ITEA 2
INFORMATION TECHNOLOGY FOR EUROPEAN ADVANCEMENT

# MEDUSA

## DELIVERABLE
## D3.1 – State of Art reports on Image Processing Services


| | |
|---|---|
| Project number: | ITEA 10004 |
| Work package: | 3 |
| Document version no.: | V1.0 final |
| Authors: | H. de Bliek, S. Delgado, F. Fahmi, C. Fetita, H. Marquering, H. Medjahed, P. Pineau, A. Papine, R. Sales Barros, S. Zeng |
| Edited by: | Henk Marquering |
| Issue date: | 14-NOV-2013 |
| Dissemination level: | Public |

**ITEA Roadmap domains:**
Major: Content & Knowledge

**ITEA Roadmap categories:**
Major: Interaction
Minor: Network & computing

**HISTORY**

| Document version # | Date | Remarks |
|---|---|---|
| V1.0 | 14-NOV-13 | First version for ITEA Board |

**TABLE OF CONTENTS**

# 1 Introduction

WP3: Image processing services is the largest work package of the MEDUSA project. In this work package it is studied how image processing services can be offered remotely and how image processing can assist clinical workflows as they are defined in WP1. This document is structured according to defined tasks in the work package: Semantic-based image compression (Section 2), Use case specific image processing and modeling (Section 3), Image Processing functionality as a Service (Section 4), Processing speed optimization (Section 5), Information integration (Section 6). Conclusions can be found in Section 7.

## 1.1 Aim of activity

The aim of WP3 is to provide advanced image processing services: (i) content-based compression of images for image size reduction in order to speed up image transfer and (ii) advanced processing to extract information from the images for the optimization of treatment decision support.

# 2 Task 3.1: Semantic-based image compression

The objective of this task is to provide efficient techniques for medical images compression to be used in the virtual platform. Basically, the compression process consists of reducing the amount of memory, storage, and transmission time required to represent data in way that this data can be recovered without any loss or without any significant loss. In the medical image context, apart from the classic and well-mastered compression approaches largely available, additional gain is expected from exploiting the semantic information contained in the medical image for the particular application to be addressed. This involves the automatic partitioning of the patient data into different knowledge levels such as patient file information, anatomic regions in the image data, and pathologic regions. This partitioning will provide a hierarchic description of the content to be compressed and shared. Because this task requires automatic isolation of different anatomic and pathologic regions in medical images, a strong connection with automatic medical images segmentation is required.

## 2.1 Medical image compression approaches

Compression techniques can be separated in two types: lossy and lossless techniques. In lossy compression techniques, the input data cannot be restored into its original. In other words, in lossy compression the reconstructed data is degraded compared to the original input data. This is done in order to achieve higher compression rates. However, because medical images are used for diagnosis purposes, degradation in the medical image data could produce unwanted visual artifacts, which is an undesired characteristic in medical applications. On the other hand, in lossless compression techniques images after compression are identical to the originals. Because of this undesired characteristic of lossy compression, this task focuses only in the usage of lossless compression techniques in medical images, applied either globally on the entire image or locally on specific regions of interest (ROIs).
Firstly, the existing lossless compression techniques that are applied to general content images will be described. These techniques focus on the removal of redundant information present in images. In addition, the existing image compression algorithms based on these techniques will be presented. The description of these algorithms will be concentrated on their application in the context of medical images. Secondly, an overview of the techniques enabling scalable lossless-to-lossy compression is presented for a purpose of ROI-based compression study. Finally, the existing approaches to isolate only useful regions of a medical image will be shown.

### 2.1.1 Techniques for redundancy removal

The existing lossless compression techniques can use one or a combination of the following approaches.

- **Run-length encoding**

Run-length encoding (RLE) is a simple compression technique based on the substitution of a sequence of consecutive data elements with the same data value by a single data value and a counter. Thus, only a single data value and a counter are stored rather than the original sequence. For instance, the sequence of data “BBBBBCCCCCAAAAAAAAAAACCC” can be stored as “5B5C11A3C”, which is shorter and requires less memory to be represented.
In order to compress 2D and 3D images, RLE can be modified to deal not only with one-dimensional sequences of data, but also to represent two-dimensional and three-

dimensional sequences with the same data value. In other words, RLE can be used to reduce the amount of data required to represent areas and volumes in medical images that have the same data value.
RLE is well suited to images that contain only a few colors, black and white images, and palette-based images. However, this compression technique is not effective for continuous-tone images. In this way, its usage in modern digital medical image is limited since there are better compression techniques.

• **Lempel–Ziv schemes**

Lempel–Ziv (LZ) schemes replace repeated occurrences of data by references to a single copy of that data. This approach differs from RLE because it is able to compress data like "CBACBACBACBA". In this case, the data "CBA" is stored in a dictionary and the input data can be represented as a sequence of 4 references to this data.
LZ approaches are also described as dictionary compression techniques because they require the construction of a dictionary of common sequences of data. LZ schemes are used in common image formats like GIF, TIFF, and PNG.

• **Huffman coding**

This compression technique is based on the statistical occurrence of pixel values. Each pixel of the image is treated as a symbol. Fewer bits are used to represent the most frequent symbols and the symbols less frequent are represent with more bits [MAS 13]. One of the characteristics of this technique is that the sequence of bits used to represent some particular symbol is never a prefix of a sequence of bits used to represent any other symbol.
This compression technique is used in several modern image compression techniques. Usually this technique is used in combination with others. The PNG format and several versions of the JPEG format use this technique.

- **Transformation coding**

In this compression technique, before being coded and stored, the images are transformed from their spatial domain into a different domain based on a well-known transform [SAH 13]. Usually, the image is transformed into the frequency or wavelet domain respectively through Fourier-related or Wavelet-related transforms. This approach is used in compression algorithms such as JPEG and JPEG 2000.
Figure 2.1 illustrates the general transform-based image compression process. First, the image pixels are decorrelated through a specific transform. Then, the result values of this transformation are quantized and these values are encoded by using LZ schemes, RLE, Huffman coding, etc.

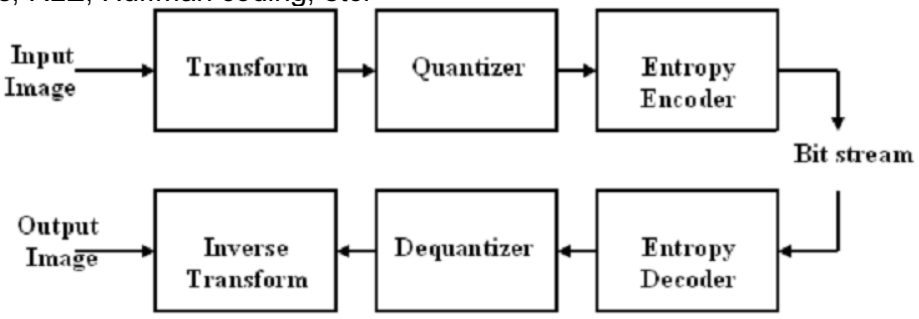


**Figure 1. Transform-based image compression (Janet et al. 2011).**

- **Predictive coding**

Predictive compression schemes are based on statistical models. These models are used to predict the value of the pixels and only the differences between the predicted value and the real pixel value are stored [SAH 13]. The quality of these compression schemes is strongly dependent of the quality of the model used to predict the pixel

value. CALIC and JPEG-LS are examples of state-of-the-art lossless compression algorithms that use this approach.

### 2.1.2 Overview of the state-of-the-art lossless image compression algorithms

Based on these approaches, several different image compression algorithms were developed. Below, the state-of-the-art algorithms used to losslessly compress medical images are described. Some of these algorithms are also used to losslessly compress general digital images. However, here they will be described and evaluated regarding their application in the context of medical images compression.

- **Deflate**

This is the compression algorithm used in the PNG image file format. This algorithm combines a LZ scheme with Huffman coding to compress the image data [KIV 98].

- **Compression based on Integer Wavelet Transforms**

A compression scheme employing classical biorthogonal filters yields floating point wavelet coefficients that have to be rounded to integers before coding. The result is that an original image cannot be perfectly reconstructed, even if lossless coding of the wavelet coefficients is performed. A reversible mapping between an integer image input and an integer wavelet representation was first offered by the sequential transform (S-transform) [HER 90]. However, the analysis and synthesis high-pass filters corresponding to this transform are of low orders (they have only one vanishing moment), yielding poor compression results, particularly for “natural” images. Higher order filters have been obtained with two similar generalizations of the S-transform: the TS transform (Zan95b) and the S+P transform [SAI 96b]. The TS transform is the integer version of the (3,1) biorthogonal wavelet transform. The S+P transform improves the compression performances of the S-transform by combining it with predictive coding. This transform has also the advantage that it can be computed with only integer addition and bit shift operations.
Different algorithms make various options in choosing the integer wavelet transform: for example, a combination between the S and TS transforms is implemented in the **CREW** (Compression with Reversible Embedded Wavelets) algorithm, the S+P transform is implemented in the algorithms of [SAI 96b], while the **SQP** algorithm [MUN 99a] implements the integer version of the interpolating transform. It is important to mention that empirical results indicate that, in lossless compression, the integer version of the interpolating transform yields the best performance for natural and medical images.

- **Compression with Reversible Embedded Wavelets**

Compression with Reversible Embedded Wavelets (CREW) is based on an integer-to-integer wavelet transform. Because of this transform, the decoding step is a lossless representation. This model works reasonably well on natural images but is not well suited for images with sharp edges [CAR 00].

- **Symmetry-Based Scalable Lossless (SBSLS) Compression of 3D Medical Image Data**

This algorithm [SAN 09] takes advantage of the anatomical symmetries present in medical images. Figure 2 illustrates the medical image symmetry and also the redundant information present in both sides of the symmetry axis. This technique uses a 2D integer wavelet transform to decorrelate the 3D image slices data. After that, a block-based prediction model is used to exploit the structural symmetry of the medical image. The residual data resulting from this prediction process are then compressed through a modified coder with optimized truncation developed according to the characteristics of the residual data. This entire process is illustrated in Figure 3.

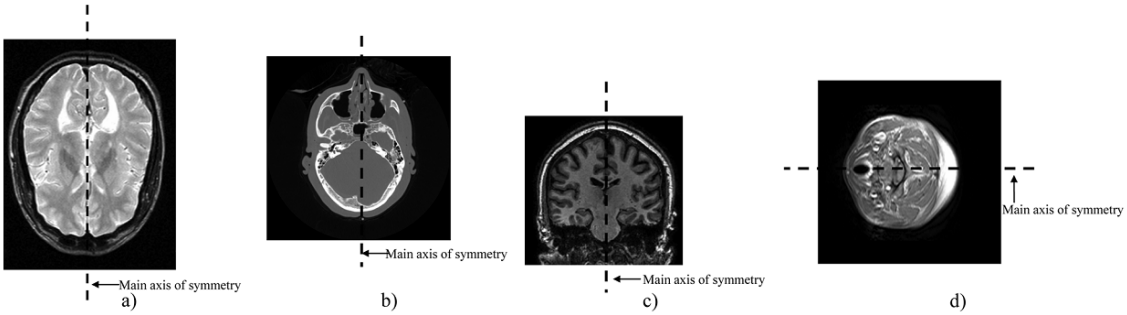


**Figure 2. Illustration of symmetry in medical images. (Sanchez et al. 2009)**

Table 2 shows the encoding and decoding time for this method, versus other compression methods discussed in the next section, namely H.264/MPEG-4 AVC, JPEG 2000, and JPEG 2000 3D. The method that uses the symmetry redundancy has the highest encoding time. However, its decoding time is similar to other compared compression methods. As can be observed in Table 3, the symmetry based compression algorithm presents an average compression ratio 15% better than other lossless methods.

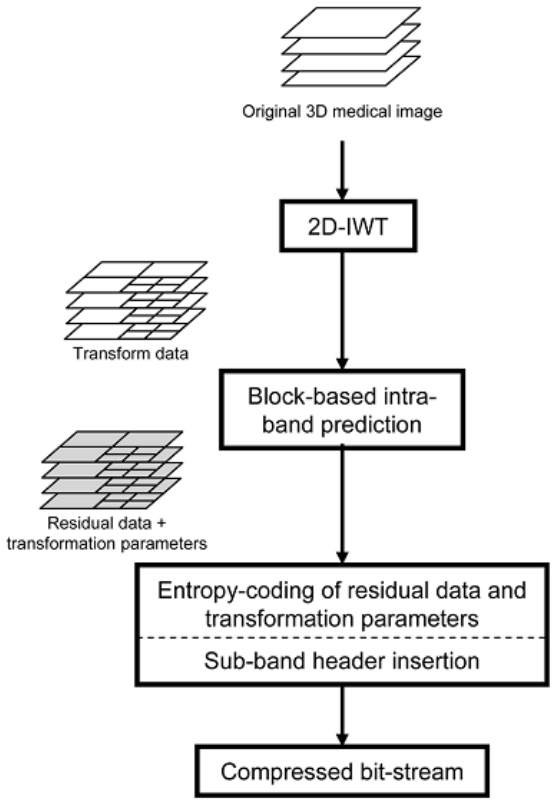


**Figure 3. Symmetry-based scalable lossless compression process**

- **JPEG-LS**

JPEG-LS is a relatively simple compression algorithm compared to alternatives with similar lossless image compression ratios. JPEG-LS is one of the compression standards incorporated in DICOM. The JPEG-LS algorithm processes the images in raster scan mode and presents two different modes of operation: "regular" and "run".

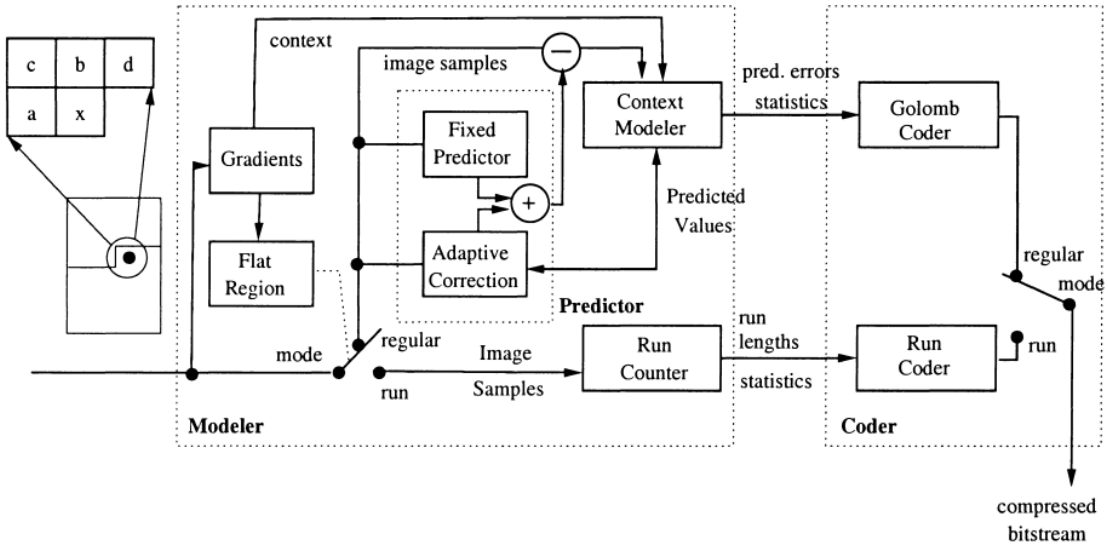


**Figure 4 JPEG-LS scheme.**

Most of the pixels are compressed in "regular" mode. In this mode, as a first step, the current pixel, which is represented by x, is predicted based on the values of the surrounding pixels a, b and c. These pixels are illustrated in Figure 4. The following non-linear predictor is used to determine the value of x:

$$\hat{x} = \begin{cases} \min(a,b) & \text{if } c \geq \max(a,b); \\ \max(a,b) & \textit{if } c \leq \min(a,b); \\ a+b-c & \text{otherwise.} \end{cases}$$

This simple predictor is able to deal properly with basic types of edges. This predictor in figure 4 is called "Fixed Predictor". JPEG-LS also provides information to a second level predictor called "Adaptive Correction" in Figure 4. The residual data or prediction errors of this second level prediction can be modeled with a two-sided geometric distribution. The remaining residual errors are them coded using the Golomb Coder. [GOL 60]

- **CALIC**

CALIC is currently the most effective lossless general content image compression method available. CALIC is a very complex algorithm but, in spite of the complexity, the memory usage of CALIC is small and it only requires the buffering of two rows of the image and some extra memory while it is running.
CALIC is also a predictive method. It starts predicting the value of the current pixel based on the previous encoded pixels of the current row and on the two previous rows. CALIC estimates the local gradient value and uses this value to select the most relevant pixels to the prediction of the current pixel. As in JPEG-LS, CALIC also has a second level of prediction that evaluates the prediction errors of the first level encoding. The resulting errors of the second level prediction are encoded used an arithmetic coder or a Huffman coder. Figure 5 presents a simplified scheme of the CALIC compression process [PHI 01].

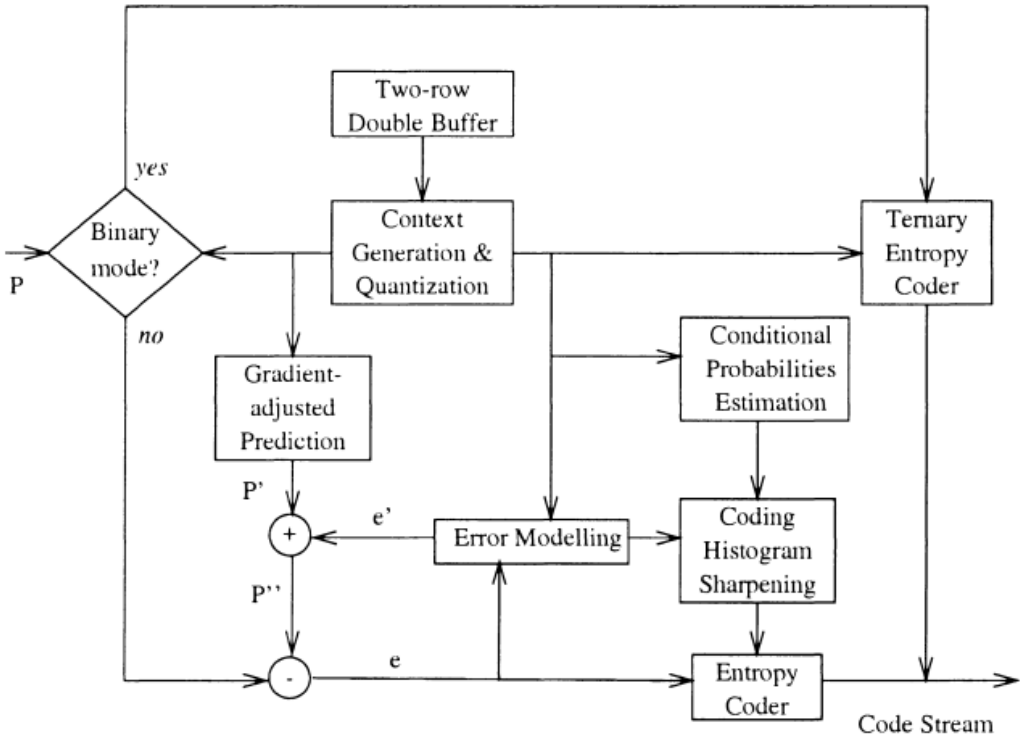


**Figure 5. CALIC simplified scheme**

### 2.1.3 Lossless-to-lossy image compression approaches

In this section we investigate the possibility to exploit lossy-to-lossless compression approaches, starting from the remark that the medical images contain most of the time a large amount of useless voxels (background) that can thus be ignored or removed. In addition, for some specific applications (or use-cases), the clinician will focus only on a given part of the body image (ROI) to make the diagnosis. This means that the remaining zones can be compressed at a lower quality. However, in the latter case, the possibility to retrieve the full image quality in degraded regions upon further request should be preserved. These functionalities require compression schemes with progressive transmission capabilities.

The ability of the wavelet-based compression techniques to create embedded data streams facilitates the progressive transmission of data over networks with limited and/or variable bandwidth: a coarse version of the image corresponding to a certain refinement level can be transmitted first, decoded, and displayed at the remote site. At this stage, the user can decide either to further refine (sometimes up to the lossless stage) the entire image or to delineate region(s) of interest to be refined first. Obviously, in order to exploit the available bandwidth optimally, the refinement information should be complementary to the previously transmitted image data, and this can only be achieved via embedded image coding. The progressive refinement of the entire image is referred to as **quality scalability**, while the second option is called **ROI coding**.
Medical workstations may have display systems of different resolutions, and, hence, the employed coding technique has to support transmission of minimal bit streams dependent on the appropriate resolution. Image database browsing should be fast and, hence, it must be feasible to retrieve low resolution versions of the database

content first. Also, in some applications (e.g., teleradiology), it might be possible that in a first instance the user is interested only in a lower resolution version of the image, and that afterwards he wants to request information about an ROI defined in the lower resolution image. Further steps can imply viewing/processing of the higher resolution versions of the image, or of the ROI, which should be transmitted progressively without any overhead. This property is addressed as **multi-resolution scalability**, and the wavelet coders can handle such a requirement in an optimal way, due to the multi-resolution nature of the wavelet transform.

- **Embedded Wavelet Coding**

Embedded coding supports progressive image transmission by the property of generating the bits in the bit stream in their relative order of importance. A given image coded at a certain bit rate in an embedded fashion stores all the lower rate codes at the beginning of the bit stream. Typically, the encoding process can be stopped before or when the target bit rate is met. Similarly, the decoder can interrupt decoding at any point in the bit stream and can reconstruct any lower rate images. Therefore, progressive transmission is supported by transmitting a coarser version of the image first, followed progressively by the refinement details. The option of lossless reconstruction is still retained, if the complete bit stream is generated, transmitted, and decoded.
Progressive image transmission via embedded wavelet coding was firstly offered by the **Embedded Zerotree Wavelet (EZW)** coding algorithm, introduced by [SHA 93]. EZW exploits the interband dependencies within the wavelet-transformed image by using efficient data models, zerotree coding, and conditional entropy coding. EZW applies successive approximation quantization (SAQ) to provide a multi-precision representation of the coefficients and to facilitate the embedded coding. SAQ uses a monotonically decreasing set of thresholds and indicates the significance of the wavelet coefficients with respect to any given threshold in a corresponding binary map (called significance map). EZW employs zero tree coding (Sha93) to encode the significance maps and prioritizes the wavelet coefficients according to their magnitude, in contrast to the typical subband coders that prioritize according to the subband frequency. In this way, EZW builds an embedded data stream, and, although optimality is not claimed, it provides graceful degradation of the image and a negligible loss in the rate-distortion (R-D) performances [SHA 93].
A more complex technique, **Set Partitioning in Hierarchical Trees (SPIHT)**, is described in [SAI 96a]. The set partitioning algorithm uses the principles of self-similarity across scales as in EZW, partial ordering by magnitude of the wavelet coefficients (resulting from SAQ), set partitioning into hierarchical trees (i.e., sorting of the trees, based on their significance, at every applied threshold), and ordered bit plane transmission of the refinement bits (i.e., the magnitude of each significant coefficient is progressively refined). The essential difference of the SPIHT coding process with respect to EZW is the way in which trees of coefficients are partitioned and sorted.
EZW and SPIHT are zerotree-based coders, and their efficiency comes from exploiting self-similarities within the wavelet transform. Coefficients from different subbands placed at the same spatial locations satisfy the zerotree hypothesis with a high probability. As a consequence, the interband redundancies are efficiently exploited by grouping the insignificant coefficients in trees growing exponentially across the scales, and by coding them with “zero” symbols (“zerotree” symbols in the terminology of [SHA 93]). However, the zero regions (corresponding to coefficients which are insignificant with respect to a given threshold) are approximated as a highly constrained set of tree structured regions; the consequence is that certain zero

regions which are not aligned with the tree structure may be expensive to code, and some portions of zero regions may not be included in zerotrees at all.
In contrast to EZW and SPIHT, the **Square Partitioning (SQP) algorithm** [MUN 99a] exploits only intraband dependencies. SQP employs a quadtreebased technique to encode the significance maps. The SQP coder abandons the tree structure constraint of its predecessors, and this prevents it from exploiting interband redundancies. However, intraband redundancies are exploited to a larger extent. The zero regions in the significance maps are represented by a union of independent square zero regions of different sizes. The coding gain resulting from an efficient exploitation of intraband redundancies compensates (at least) for the losses incurred by not using an interband decorrelation technique. The lossless compression performances [MUN 99a] report on multimodal medical images reveal that SQP provides competitive results, similar to the state-of-the-art CALIC technique [WU 97]. Moreover, lossy compressions of typical photographic images indicate that the SQP's performances are similar to SPIHT [SAI 96a].
An alternative quadtree coding approach algorithm, **the Wavelet QuadTree (WQT)** [MUN 99b] implemented a similar approach as in SQP, but with the difference that it introduces the possibility of multi-ROI coding: the end user receives an image which is refined up to a certain level and has the possibility to delineate one or several ROIs in the decoded image for which he requires a lossless reconstruction.

- **JPEG 2000**

JPEG 2000 is one of the modern compression standards incorporated into DICOM. This image compression algorithm was developed to provide better compression ratio when compared with the traditional JPEG algorithm. JPEG 2000 also presents other useful features:

- multiple resolution representation: JPEG 2000 was developed in such a manner that it decomposes the image in different resolutions which can be used in specific non-compression related applications.
- progressive transmission: JPEG 2000 enables the progressive transmission of the entire image data in a way that makes it possible to visualize lower quality versions of the final image while the image data is being transmitted;
- lossless compression: the original JPEG algorithm does not allow image compression without data degradation, however, differently from JPEG, JPEG 2000 has support to lossy and lossless compression;
- error resilience: JPEG has native mechanisms to deal with errors introduced by data transmission or storage devices;
- regions of interest: JPEG 2000 permits spatial random access or region of interest access to the image data in different levels of granularity and this enables storage of different regions of an image using different compression quality levels;
- JPEG 2000 has the capacity to store more than 8 bits per channel images, which is particularly useful for medical applications where high dynamic range images are common.

JPEG 2000 is classified as a transformation compression technique based on a wavelet transform.

- **JPEG 2000 3D**

JPEG 2000 3D is an extension of JPEG 2000 to deal with the compression of 3D images. JPEG 2000 3D uses a discrete wavelet transform across the 3D image slices and the result of this transformation is encoded using JPEG 2000 [SAN 09].

- **H.264/MPEG-4 AVC**

This algorithm was specifically designed for video compression. However, it can also be applied to compress images. Here, a 3D medical image is interpreted as a video, in which each slice is considered as a frame of a video. By interpreting the medical images as videos it is possible to use advanced video compression algorithms to take advantage of the inter-slice redundant data. This algorithm is an international standard for video compression and it is based on multiframe motion compensation and estimation using blocks with variable size. It also has the possibility to deal with macro-block quality refinement. However, it cannot achieve lossless compression, even if high-quality image can be obtained after decoding.

### 2.1.4 Comparison of compression performance

As we can see in Table 1, CALIC has the best compression ratio in general. Calic only has an inferior compression ratio when compared to JPEG 2000 regarding medical images with 8 or less bits per pixel. To calculate the data displayed in Table 5, 3679 medical images of different modalities (CT, MRI, US, etc.) were compressed using different algorithms. Also regarding the data in Table 5, JPEG-LS present a satisfying compression ratio and with a very fast encoding time.

**Table 1 Comparing the compression rates different lossless image compression algorithms applied to medical images.**

| Compression Scheme | All images (3679) | Images with 8 or less bits per pixel (731) | Images with more than 8 bits per pixels (2948) |
|---|---|---|---|
| DEFLATE (PNG) | 2.76 | 3.31 | 2.62 |
| CREW | 3.56 | 3.32 | 3.6 |
| JPEG | 3.04 | 2.62 | 3.14 |
| JPEG-LS | 3.81 | 3.86 | 3.8 |
| JPEG 2000 | 3.81 | 4.11 | 3.74 |
| CALIC | 3.91 | 3.93 | 3.9 |

**Table 2 Encoding and decoding time of the SBSLS method [SAN 09].**

| **Modality** *slices: pixels per slice: bits per pixel* | **Compression method** | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | H.264/AVC intra-coding | | JPEG2000 | | 3D-JPEG2000 | | **Proposed method** | |
| | EcT | DcT | EcT | DcT | EcT | DcT | EcT | DcT |
| 1.MRI: *24:192×256:16* | 43.51 | 3.46 | 4.01 | 3.18 | 4.10 | 3.18 | **48.23** | **3.23** |
| 2.MRI: *35:256×256:16* | 49.77 | 3.59 | 4.43 | 3.55 | 4.48 | 3.57 | **56.12** | **3.74** |
| 3.MRI: *11:512×512:8* | 88.85 | 5.27 | 6.61 | 5.23 | 6.63 | 5.26 | **95.12** | **5.24** |
| 4.CT: *40:512×512:16* | 321.36 | 18.75 | 24.30 | 18.82 | 24.31 | 18.83 | **335.12** | **18.84** |
| 5.CT: *40:512×512:16* | 322.10 | 18.78 | 24.06 | 18.99 | 24.09 | 19.00 | **349.67** | **19.06** |

EcT: encoding time. DcT: decoding time.
MRI: magnetic resonance imaging. CT: computed tomography.

**Table 3 Comparing the compression ratio of the symmetry-based method.**

| **Modality** *slices: pixels per slice: bits per pixel* | **Compression method** | | | |
|---|---|---|---|---|
| | H.264/AVC intra-coding | JPEG2000 | 3D-JPEG2000 | **Proposed method** |
| | compression ratio (bit-rate: bits per pixel) | | | |
| 1. MRI 24:192×256:16 | 2.65:1 (6.03 bpp) | 2.62:1 (6.10 bpp) | 2.71:1 (5.90 bpp) | **3.25:1 (4.92 bpp)** |
| 2. MRI 30:192×256:16 | 2.58:1 (6.20 bpp) | 2.51:1 (6.33 bpp) | 2.68:1 (5.97 bpp) | **3.21:1 (4.98 bpp)** |
| 3. MRI 30:256×256:16 | 2.41:1 (6.63 bpp) | 2.57:1 (6.22 bpp) | 2.69:1 (5.94 bpp) | **3.28:1 (4.87 bpp)** |
| 4. MRI 35:256×256:16 | 2.41:1 (6.63 bpp) | 2.57:1 (6.22 bpp) | 2.68:1 (5.97 bpp) | **3.29:1 (4.86 bpp)** |
| 5. MRI 60:256×256:16 | 3.31:1 (4.83 bpp) | 3.48:1 (4.59 bpp) | 3.52:1 (4.54 bpp) | **3.86:1 (4.14 bpp)** |
| 6. MRI 55:256×256:16 | 3.29:1 (4.86 bpp) | 3.45:1 (4.63 bpp) | 3.48:1 (4.59 bpp) | **4.01:1 (3.99 bpp)** |
| 7. MRI 65:224×192:16 | 2.31:1 (6.92 bpp) | 2.61:1 (6.13 bpp) | 2.64:1 (6.06 bpp) | **3.03:1 (5.28 bpp)** |
| 8. MRI 58:224×192:16 | 2.33:1 (6.86 bpp) | 2.58:1 (6.20 bpp) | 2.61:1 (6.13 bpp) | **2.92:1 (5.47 bpp)** |
| 9. MRI 24:256×256:16 | 8.58:1 (1.86 bpp) | 8.45:1 (1.89 bpp) | 8.49:1 (1.88 bpp) | **9.49:1 (1.69 bpp)** |
| 10. MRI 28:256×256:16 | 8.61:1 (1.85 bpp) | 8.46:1 (1.89 bpp) | 8.50:1 (1.88 bpp) | **9.52:1 (1.68 bpp)** |
| 11. MRI 26:256×256:16 | 8.48:1 (1.88 bpp) | 8.42:1 (1.90 bpp) | 8.47:1 (1.89 bpp) | **9.44:1 (1.69 bpp)** |
| 12. MRI 24:256×256:16 | 8.27:1 (1.93 bpp) | 8.21:1 (1.94 bpp) | 8.29:1 (1.93 bpp) | **9.20:1 (1.73 bpp)** |
| 13. MRI 182:176×176:16 | 3.23:1 (4.95 bpp) | 3.41:1 (4.69 bpp) | 3.44:1 (4.65 bpp) | **3.75:1 (4.26 bpp)** |
| 14. MRI 190:176×176:16 | 3.37:1 (4.74 bpp) | 3.51:1 (4.55 bpp) | 3.65:1 (4.38 bpp) | **3.87:1 (4.13 bpp)** |
| 15. MRI 180:176×176:16 | 3.27:1 (4.89 bpp) | 3.37:1 (4.74 bpp) | 3.58:1 (4.46 bpp) | **3.70:1 (4.32 bpp)** |
| 16. MRI 196:176×176:16 | 3.15:1 (5.07 bpp) | 3.18:1 (5.03 bpp) | 3.37:1 (4.74 bpp) | **3.65:1 (4.38 bpp)** |
| 17. MRI 11:512×512:8 | 2.85:1 (2.80 bpp) | 2.84:1 (2.81 bpp) | 2.88:1 (2.77 bpp) | **3.10:1 (2.58 bpp)** |
| 18. MRI 50:512×512:8 | 2.93:1 (2.73 bpp) | 2.91:1 (2.74 bpp) | 2.96:1 (2.70 bpp) | **3.18:1 (2.51 bpp)** |
| 19. CT 40:512×512:16 | 3.89:1 (4.11 bpp) | 4.44:1 (3.60 bpp) | 4.51:1 (3.54 bpp) | **5.34:1 (2.99 bpp)** |
| 20. CT 40:512×512:16 | 3.81:1 (4.19 bpp) | 4.39:1 (3.64 bpp) | 4.49:1 (3.56 bpp) | **5.28:1 (3.03 bpp)** |
| 21. CT 40:512×512:16 | 3.52:1 (4.92 bpp) | 4.16:1 (3.84 bpp) | 4.28:1 (3.73 bpp) | **4.50:1 (3.55 bpp)** |
| 22. CT 40:512×512:16 | 3.68:1 (4.34 bpp) | 4.12:1 (3.88 bpp) | 4.19:1 (3.81 bpp) | **5.09:1 (3.14 bpp)** |

MRI: magnetic resonance imaging. CT: computed tomography

### 2.1.5 Techniques for useless voxels removal

Compression schemes can have high compression rates when quality loss can be afforded. However, deficiencies cannot be allowed in regions that are important. A common idea is to preserve quality in diagnostically critically regions while allowing lossy encoding in others. In multi-trauma all regions of a patient may be important, whilst in stroke patients the whole brain can be considered as the region of interest. Thus, we need to partition a medical image into two sets of pixels/voxels, the useful and the useless. This partitioning is called image segmentation. In other words, a segmentation process must be realized in a medical image in order to select the useful voxels in a medical image. Image segmentation is a wide topic and there are many techniques to segment images. Also, depending on the quality of the segmentation result and on the purpose of the segmentation process, the processing time required by these techniques varies significantly.
The objective of MEDUSA project is to deal with time-critical medical situations. Therefore, a fast execution is important. Thus, we here concentrate on fast medical images segmentation techniques. As a result, it is possible that coarse or imprecise segmentation is achieved. However if this segmentation is able to reduce unused parts of the medical image, such as the regions outside the human body, then this segmentation is suited for this project. Also, by removing this useless image data it is possible to speed up additional image processing algorithms because this removed data will not be processed. In addition, to reduce the time required by the segmentation process, fully automated techniques or techniques that do not strongly dependent on user input are more appropriate because they eliminate user interaction time of the process. Because of this, only a few segmentation techniques are suited for MEDUSA.
Moreover, the same segmentation technique can be implemented to run in different types of hardware. [PIE 13] compared the execution time of the same segmentation technique running in three different hardware platforms: CPU, GPU and FPGA. This showed that GPU has the best performance in execution time (see Table 4) and in data transfer time (see Table 5). The effective development of solutions based on FPGA technology is a complex task. Also, the FPGA hardware is expensive and not easily accessible compared to the CPU and GPU. Thus, the following work does not concentrate in the usage of FPGA technology.

**Table 4 Segmentation execution time.**

| Number of pixels | # GPU [ms] | # CPU [ms] | # FPGA [ms] |
|---|---|---|---|
| 2048 | 0,6 | 1,82 | 0,16896 |
| 10240 | 0,74 | 36,1 | 0,8448 |
| 51200 | 1,58 | 180,7 | 4,224 |
| 204800 | 5,13 | 714 | 16,896 |
| 512000 | 12,15 | 1813,54 | 42,24 |

**Table 5 Segmentation data transfer time.**

| Number of pixels | # GPU [ms] | # FPGA [ms] |
|---|---|---|
| 2048 | 0,45 | 1,12 |
| 10240 | 0,49 | 1,22 |
| 51200 | 0,66 | 1,65 |
| 204800 | 1,3 | 3,25 |
| 512000 | 2,6 | 6,5 |

- **Support Vector Machine Classifier**

Support vector machines (SVM) are used for binary classification of data, which means that they separate the data in two classes. SVM can be interpreted as process to determine a hyperplane which divides the input data in an n-dimensional space. [PIE13] uses the SVM algorithm implementing image segmentation process in 3 different hardware platforms and it produces results that are fast enough for real-time segmentation in a GPU. In Table 4 and in Table 5 the results of this work can be observed.

- **Quick Shift Algorithm**

Quick shift is a kernel-based segmentation algorithm that can be applied to segment data in any dimensional space. [FUL 12] evaluate the performance of a GPU implementation of the quick shift algorithm. This work applies the segmentation process to RGB images and the algorithm evaluates the 3 color components of a pixel augmented with its two position coordinates. Thus, the algorithm is processing five dimensional data. This work found that this GPU implementation can segment around 10 images of 256×256 pixels per second. An example of image segmentation result can be observed in Figure 6.

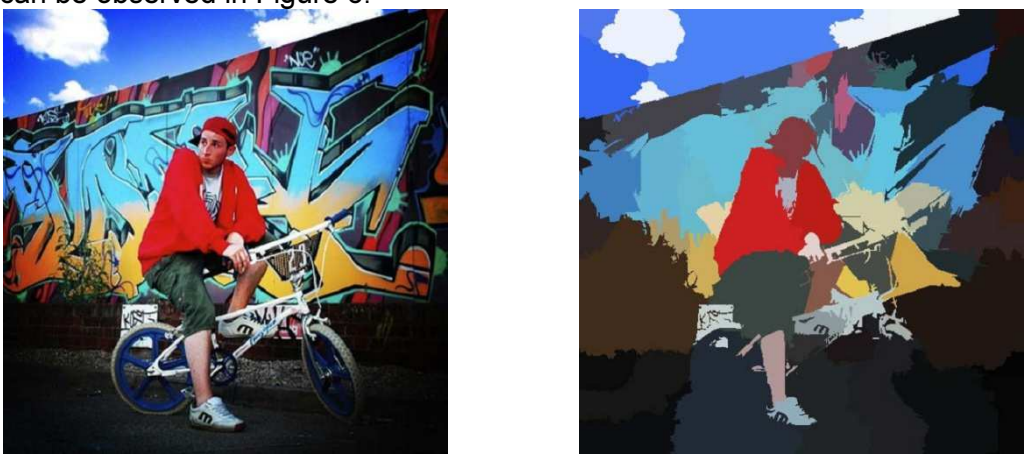

**Figure 6. Example of quick shift segmentation result.**

- **Chan-Vese Model**

Chan-Vese model is an active contour model. This model aims to determine object outlines in noise 2D images. [LI 13] propose a different implementation of this algorithm that is well suited to medical images and presents a best performance compared to the traditional implementation of this model. In this work, a CT scan was segmented in 0.0406 seconds. In Figure 7 the segmentation result of this implementation can be observed.

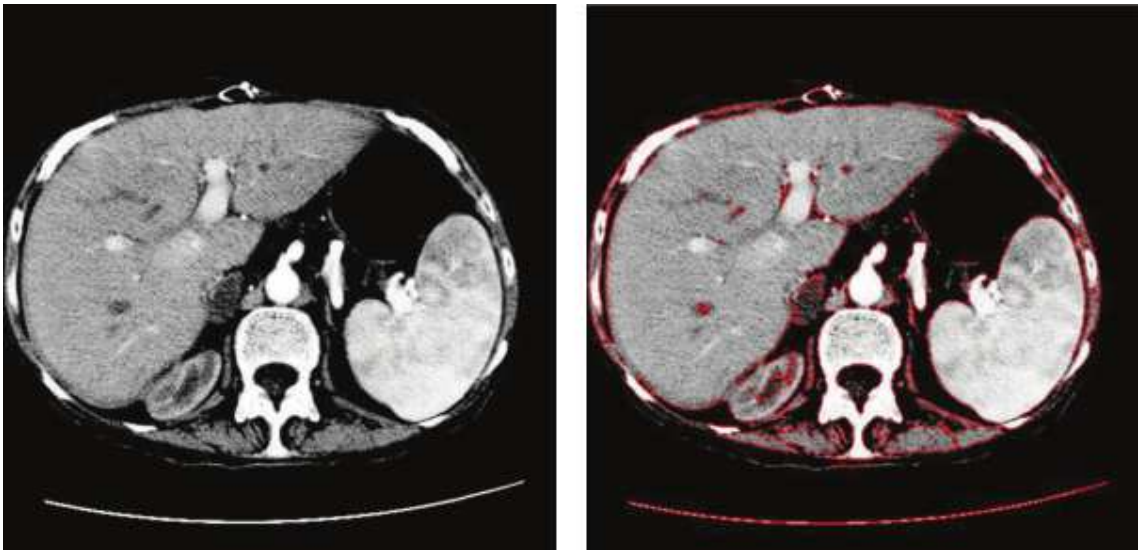

**Figure 7. Chan-Vese segmentation result.**

- **Clustering Approaches**

Clustering algorithms are called unsupervised classification methods that organize unlabeled feature vectors into clusters. The samples in a cluster are more similar to each other than the samples belonging to different clusters. Clustering algorithms are extensively explored for faster medical image segmentations.
[BAL 12] uses the Lattice Boltzmann method (LBM) for solving partial differential equations together with the fuzzy c-means clustering algorithm to segment real-world images, which include medical images. Using the gradient descent method, the corresponding level set equation is obtained and from this equation a fuzzy external force is deduced and solved by the LBM.
[RAS 12] tries to develop a best segmentation algorithm by using a kernel-based fuzzy c-means (KFCM) clustering algorithm. The approach proposed in this work is defined by two successive stages for image segmentation. First, the KFCM is used to cluster the input image. After that, the region of interest fuzzy membership matrix is used in the next stage as an initial contour. Then, the fast two cycle model is utilized to segment the image by curve evolution.
[SRI 10] proposes an algorithm for segmentation of brain MR images for clinical use that is noise adaptive. This work presents a hybrid clustering algorithm which integrates the concepts of rough sets and fuzzy sets. The approximations of rough sets are incorporated to handle uncertainty, vagueness, and incompleteness. The images are proposed to be pre-processed with a neighborhood averaging spatial filter for making the segmentation robust to noise.
[XIE 10] establishes a strategy to take benefit of the Gaussian mixture models (GMMs) in the segmentation of medical images. In this context, the estimation of the number of components is still an open question. Also, the speed of the GMM's for large medical image is slow, and GMMs noise sensitivity is high. [XIE 10] uses the kernel density estimation method to approximate the number of components K, and proposes three strategies to speed up the GMMs segmentation. First, a histogram stratification sampling strategy is projected to reduce the training data. Next, a binning strategy is designed to search the neighbor points of each center data to compute the approximate density function of the samples. Finally, a hill-climbing algorithm with dynamic step size is proposed to find the local maxima of the density function. The kernel density estimation method and sampling technology are used to reduce the noise present in the images.

[GIB 02] establishes a relationship between a level set algorithm and k-means algorithm and exploits this relationship to develop a new hybrid numerical technique for segmentation that draws on the speed and simplicity of k-Means procedures, and the robustness of level set algorithms.
All the clustering algorithms described in this section are able to segment a medical image requiring no longer than one second.

- **Other Alternatives:**

Compression is about reducing file size by removing redundancy. There are different redundancy types such as spatial and temporal as described before. Remarkably, medical image data in the DICOM data format also has a lot of redundancy of meta information in the header. Typically every slice is stored in a separate file and each file contains meta information like the name of a patient, date of birth. Little is known about how much the data can be compressed by omitting this information in every slice. For example, some vendors allow the storage of the whole volume in a single file.

## 2.2 Data transfer for the Use Case "Contouring for lymphoma"

The use case "Contouring for lymphoma", from the user story "Contouring for Oncology", involves several image series from different modalities for a distant and collaborative work between many actors. Each anatomical or functional series consists of 3D volume of large data. This use case represents the worst case regarding data capacity for this user story.
An example is detailed below to get an idea of the total amount of data the use case asks to manage.
At the beginning of the use case, the actor loads 3 series: $CT0_{RT}$, CT0, and PT0, which are located within a distant database.
For a standard case, the amount of data without any compression of each series is summarized in Table 6.

**Table 6 – Data sizes for standard case (lymphoma)**

| Series | Type of data | Approximate size without any compression |
|---|---|---|
| $CT0_{RT}$ | Enhanced CT series performed before chemotherapy<br>May be limited to one part of the body | 512x512x256 slices x 2 Bytes / voxel = 128 MB |
| CT0 | CT series from hybrid PET scan<br>Whole body acquisition | 512x512x800 slices x 2 Bytes / voxel = 400 MB |
| PT0 | PT series from hybrid PET scan<br>Whole body acquisition | 256x256x320 slices x 2 Bytes / voxel = 40 MB |
| ***All*** | | ***Around 500 MB*** |

Going further to the use case, the actor performs the co-registration of CT0 and PT0 to $CT0_{RT}$, the new registered series are named $CT0_R$ and $PT0_R$. Thereby, 2 new series are added to the current study context. The data sizes at this point are presented in Table 7– an overview is presented in Figure 8.

**Table 7 – Data sizes for lymphoma case (pre-chemo).**

| Series | Type of data | Approximate size without any compression |
|---|---|---|
| $CT0_{RT}$ | Enhanced CT series performed before chemotherapy<br>May be limited to one part of the body | 512x512x256 slices x 2 bytes / voxel = 128 MB |
| CT0 | CT series from hybrid PET scan<br>Whole body acquisition | 512x512x800 slices x 2 bytes / voxel = 400 MB |
| PT0 | PT series from hybrid PET scan<br>Whole body acquisition | 256x256x320 slices x 2 bytes / voxel = 40 MB |
| $CT0_R$ | CT0 registered to $CT0_{RT}$ | 512x512x800 slices x 2 bytes / voxel = 400 MB |
| $PT0_R$ | PT0 registered to $CT0_{RT}$ | 256x256x320 slices x 2 bytes / voxel = 40 MB |
| ***All*** | | ***Around 1 GB*** |

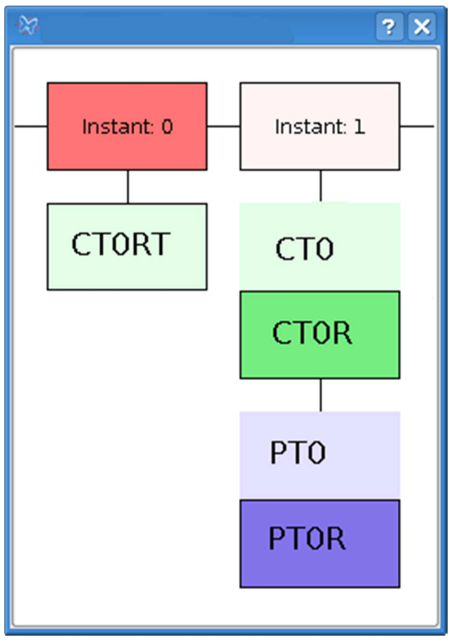


**Figure 8. Pre chemo series data tree**

After the chemotherapy treatment, the actor creates a new study including post chemo series: $CT1_{RT}$, CT1 and PT1 and performs the co-registration of CT1 and PT1 to $CT1_{RT}$ in order to get $PT1_R$ and $CT1_R$. He will also perform the co-registration of

$CT0_{RT}$ to $CT1_{RT}$ ($CT0_{RTR}$) and get at the end the following series data tree and the following data table taken into account multiple points and registered volumes to baseline. This configuration is required to be able to perform, in the same coordinate system, the contours on the relevant series $CT1_{RT}$, $PT1_{R}$ and $CT0_{RTR}$.

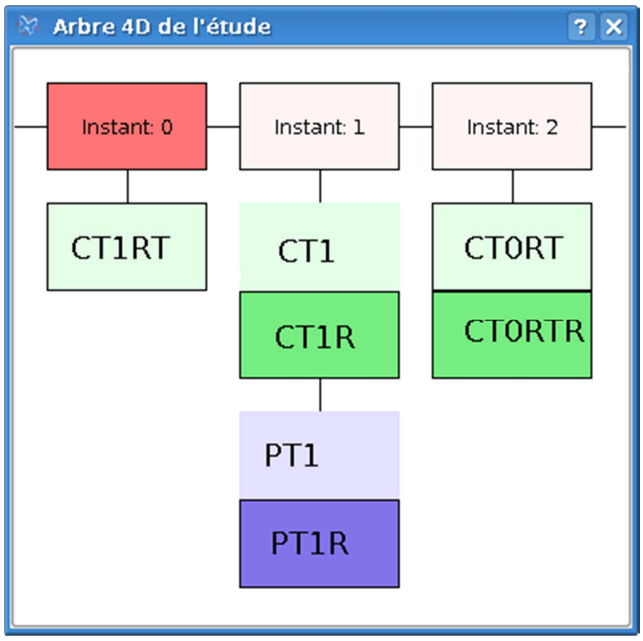


**Figure 9. Post chemo series data tree**

**Table 8. Data sizes post-chemo0 (lymphoma case)**

| Series | Type of data | Approximate size without any compression |
|---|---|---|
| $CT1_{RT}$ | Enhanced CT series performed after chemotherapy<br>May be limited to one part of the body | 512x512x256 slices x 2 bytes / voxel = 128 MB |
| CT1 | CT series (Post chemo) from hybrid PET scan<br>Whole body acquisition | 512x512x800 slices x 2 bytes / voxel = 400 MB |
| PT1 | PT series (Post chemo) from hybrid PET scan<br>Whole body acquisition | 256x256x320 slices x 2 bytes / voxel = 40 MB |
| $CT1_{R}$ | CT1 registered to $CT1_{RT}$ | 512x512x800 slices x 2 bytes / voxel = 400 MB |
| $PT1_{R}$ | PT1 registered to $CT1_{RT}$ | 256x256x320 slices x 2 bytes / voxel = 40 MB |
| $CT0_{RT}$ | Enhanced CT series performed before chemotherapy<br>May be limited to one part of the body | 512x512x256 slices x 2 bytes / voxel = 128 MB |
| $CT0_{RTR}$ | $CT0_{RT}$ registered to $CT1_{RT}$ | 512x512x256 slices x 2 bytes / voxel = 128 MB |
| *All* | | Around 1.2 GB |

The total capacity of the data to be transferred and managed in a study will be generally greater than 1 GB, for the use case "Contouring for lymphoma" when no compression has been applied.

## 2.3 References

[BAL 12] Balla-Arabé, S., Gao, X. & Wang, B., 2012. A Fast and Robust Level Set Method for Image Segmentation Using Fuzzy Clustering and Lattice Boltzmann Method. IEEE transactions on systems, man, and cybernetics. Part B, Cybernetics : a publication of the IEEE Systems, Man, and Cybernetics Society, 43(3), pp.910–920. Available at: http://www.ncbi.nlm.nih.gov/pubmed/23076068.

[CAR 00] Carpentieri, B., Weinberger, M.J. & Seroussi, G., 2000. Lossless Compression of Continuous-Tone Images. In Proceedings of the IEEE. IEEE, pp. 1797–1809.

[FUL 12] Fulkerson, B. & Soatto, S., 2012. Really Quick Shift : Image Segmentation on a GPU. , i, pp.350–358.

[GIB 02] Gibou, F. & Fedkiw, R., 2002. A Fast Hybrid k-Means Level Set Algorithm For Segmentation. In 4th Annual Hawaii International Conference on Statistics and Mathematics. pp. 281–291.

[GOL 60] http://en.wikipedia.org/wiki/Golomb_coding

[HER 90] V. K. Heer and H.-E. Reinfelder, “A comparison of reversible methods for data compression,” in Proc. SPIE, Medical Imaging IV, 1990, vol. 1233, pp. 354–365.

[JAN 11] Janet, J., Mhandass, D. & Meenalosini, S., 2011. Lossless Compression Techniques for Medical Images in Telemedicine. In G. Graschew, ed. Advances in Telemedicine: Technologies, Enabling Factors and Scenarios. pp. 111–130.

[KIV 98] Kivijärvi, J. et al., 1998. A comparison of lossless compression methods for medical images. Computerized Medical Imaging and Graphics, 22(4), pp.323–339.

[LI 13] Li, S. & Zhang, Q., 2013. Fast Image Segmentation Based on Efficient Implementation of the Chan-Vese Model with Discrete Gray Level Sets. Mathematical Problems in Engineering, 2013, pp.1–16.

[MAS 13] Masood, S. et al., 2013. Brain Image Compression : A Brief Survey. Research Journal of Applied Sciences, Engineering and Technology, 5(1), pp.49–59.

[MUN 99a] A. Munteanu, J. Cornelis, G. Van der Auwera, and P. Cristea, “A wavelet based lossless compression scheme with progressive transmission capability,” Int. J. Imaging Syst. Technol., vol. 10, pp. 76–85, 1999.

[MUN 99b] Munteanu, A.; Cornelis, J.; Van der Auwera, G.; Cristea, P., "Wavelet image compression - the quadtree coding approach," Information Technology in Biomedicine, IEEE Transactions on , vol.3, no.3, pp.176,185, Sept. 1999

[PHI 01] Philips, W. et al., 2001. State-of-the-art techniques for lossless compression of 3D medical image sets. Computerized Medical Imaging and Graphicss, 25(2), pp.173–185.

[PIE 13] Pietron, M. et al., 2013. Comparison of GPU and FPGA Implementation of SVM Algorithm for Fast Image Segmentation. In H. Kubátová et al., eds. Architecture of Computing Systems – ARCS 2013. Prague, Czech Republic: Springer-Verlag Berlin Heidelberg, pp. 292–302.

[RAS 12] Rastgarpour, M., Alipour, S. & Shanbehzadeh, J., 2012. Improved Fast Two Cycle by using KFCM Clustering for Image Segmentation. , I.

[SAH 13] Sahu, N.K. & Kamargaonkar, C., 2013. A SURVEY ON VARIOUS MEDICAL IMAGE COMPRESSION TECHNIQUES. International Journal of Science, Engineering and Technology Research, 2(2), pp.501–506.

[SAI 96a] A. Said and W. Pearlman, “A new fast and efficient image codec based on set partitioning in hierarchical trees,” IEEE Trans. Circuits Syst. Video Technol., vol. 6, pp. 243–250, 1996.

[SAI 96b] A. Said and W. Pearlman, “An image multiresolution representation for lossless and lossy compression,” IEEE Trans. Image Processing, vol. 5, pp. 1303–1310, 1996.

[SAN 09] Sanchez, V., Abugharbieh, R. & Nasiopoulos, P., 2009. Symmetry-Based Scalable Lossless Compression of 3D Medical Image Data. IEEE Transactions on Medical Imaging, 28(7), pp.1062–1072.

[SHA 93] J. M. Shapiro, “Embedded image coding using zerotrees of wavelet coefficients,” IEEE Trans. Signal Processing, vol. 41, pp. 3445–3462, 1993.

[SRI 10] Srivastava, A., Asati, A. & Bhattacharya, M., 2010. A Fast and Noise-Adaptive Rough-Fuzzy Hybrid Algorithm for Medical Image Segmentation. 2010 IEEE International Conference on Bioinformatics and Biomedicine Medical Image Segmentation, pp.416–421.

[WU 97] X. Wu, “Lossless compression of continuous-tone images via context selection, quantization, and modeling,” IEEE Trans. Image Processing, vol. 6, pp. 656–664, 1997.

[XIE 10] Xie, C.-H., Song, Y.-Q. & Chen, J.-M., 2010. Fast medical image mixture density clustering segmentation using stratification sampling and kernel density estimation. Signal, Image and Video Processing, 5(2), pp.257–267. Available at: http://link.springer.com/10.1007/s11760-010-0159-7 [Accessed May 15, 2013].

# 3 Task 3.2: Use case specific image processing and modeling

The Medusa platform will be used for various medical demonstrators. For the demonstrators computationally demanding image processing tooling is required. This chapter introduces the various scenarios of the medical demonstrators and describes the current state of the art of image processing functionality for these clinical subjects. This chapter describes the state of the art for stroke, multi-trauma, and oncology.

## 3.1 Stroke oriented image processing

Stroke is a leading cause of death and the main cause of serious long term disability. Traditionally, stroke patients were not imaged extensively because there were not many treatment options. If an ischemic stroke patient arrives within 3 hours after stroke onset, thrombolysis could be administered and otherwise conservative therapy was performed. The only imaging that traditionally was performed was a non-contrast CT to distinguish a hemorrhagic stroke from an ischemic stroke. The evaluation of these images is required before the initiation of therapy with tissue plasminogen activator. Because this distinction is relatively easy on these images, not much effort has been put in any automated analysis up to date. With the advent of new treatment methods, image based patient selection has become important, which opens up the possibility of computer aided support of the interpretation of the images.

### 3.1.1 Current analysis

Medical imaging techniques play an increasingly important role in the diagnostic process of hyperacute stroke. Currently, a patient with acute ischemic stroke is imaged almost exclusively with CT. The following images are generated:

- Baseline non-contrast CT. This scan is always performed, mainly for the distinction between hemorrhagic and ischemic stroke. For stroke patients this scan is used to:
    - Detect hemorrhages;
    - Detect early ischemic changes;
    - Detect high density signs.
- Baseline CT Angiography. In this scan protocol contrast agent is administered to make the arteries visible. In the patient work up it is used to:
    - Detect the presence of thrombus that occludes the intracranial artery;
    - Detect thrombus dimensions, and intensity;
    - Detect collateral circulation;
    - Detect early ischemic changes;
- Baseline CT Perfusion. In this scan protocol multiple images are acquired during a time period. It is used to:
    - Detect infarcted core and penumbra;
    - Detect co-lateral circulation;
    - Quantify thrombus dimensions;
    - Quantify permeability;
- Follow-Up CT: between 5-7 days an additional CT is made to
    - Quantify infarcted core size;
- Follow-Up CTA: These scans are rarely made and only for research purposes. These scans can be used to:
    - Quantify collateral circulation;

- Follow-Up CT Perfusion: These scans are rarely made and only for research purposes. These scans can be used to:
    - Quantify infarcted core;
    - Quantify collateral circulation;
- DSA and 3DRA: imaging during intravascular interventions. These scans can be used to:
    - Navigate;
    - Quantify collateral circulation;
    - Evaluate the success of the treatment;
- Diffusion Weighted MRI imaging: not commonly used because of the absence of available MR scanners in acute situations.
    - Quantification of infarct core;

The amount of image data is quite large and can be a couple of Gigabytes.
Various initiatives have been started to develop automated scoring systems to come to image-based decision support systems for acute stroke care. Below, the state of the art of image processing of these images is described.

### 3.1.2 Automatic hemorrhage detection

We found only one publication regarding the automatic detection of intracerebral hemorrhage. [SHA 12] uses the K-means clustering algorithm to segment the hemorrhage in non-contrast CT scans. In this algorithm, the centers of the clusters have to be initialized. Subsequently, the hemorrhages are detected automatically using a method that analyzes the peaks image histograms. After this analysis, the K-cluster algorithm is executed and it defines 3 clusters in the images. (See Figure 10) This figure shows the biconvex shape with the brightest pixels representing the segmentation of an epidural hemorrhage. The quality of this method is sensitive to various artifacts commonly present in CT images. Because of this sensibility to common CT images artifacts, this method is not very robust.

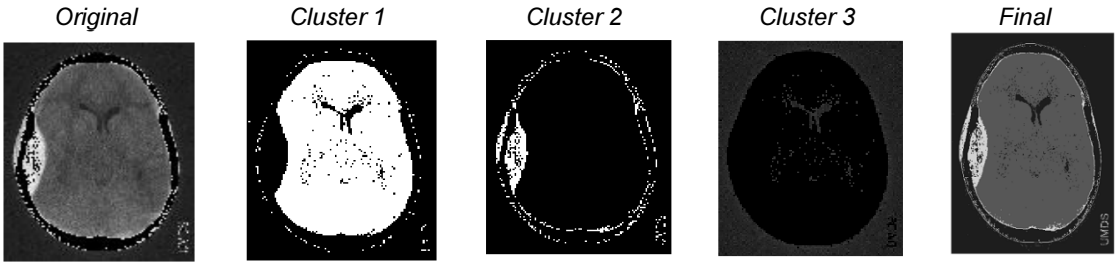


**Figure 10. Clusters of original image.**

### 3.1.3 Automatic early infarct changes detection

The anatomical changes in acute infarction cases can be subtle in the first hours after the onset of symptoms. Some automatic techniques were developed to detect these early changes.
Maldjean et al. [MAL 01] developed a method to identify potential areas of acute ischemia on CT scans. In this method, all image information outside the brain (including the skull) is removed. Next, the remaining data is matched to a standard atlas and segmented into anatomic regions. Finally, the intensities of the voxels are compared with the contralateral side in order to identify areas of early ischemic changes.

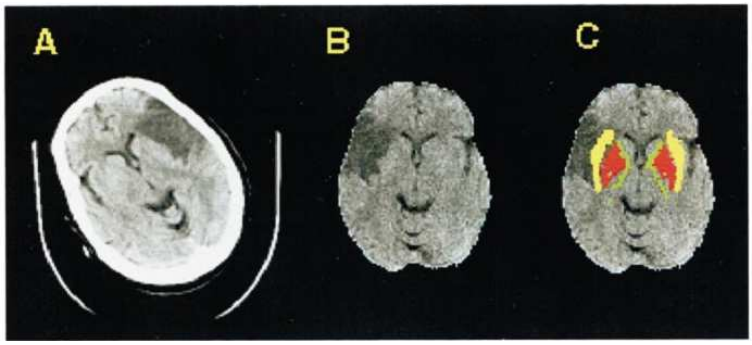


**Figure 11. A, original CT scan data. B, CT scan after extraneous features removal and normalization. C, automated segmentation.**

A similar method is described in [SHI 12]. [SHI 12] also removes the skull information of the analyzed CT scans and realigns this image data with a standard brain atlas. Next, the boundaries of ten specific regions of interest are automatically determined. Subsequently, a 3-means clustering is done and the intensity of the labeled voxels is compared with the contralateral side. Differences exceeding a specific threshold are used for statistical analysis and scoring.
[KOS 10] proposes to automate an MR topographical score (MR-TS) using a digital atlas to develop an objective tool for large-scale analyses and possibly reduce interrater variability and slice orientation differences. An MR-TS digital atlas was generated using the ASPECTS[1] scoring sheet and anatomic MR data sets. Automated MR topographical scores (auto-MR-TS) were obtained based on the overlap of lesions on apparent diffusion coefficient maps with MR-TS atlas regions. Auto-MR-TS provides a measure of stroke severity in an automated fashion and facilitates more objective, sensitive, and potentially more complex ASPECTS-based scoring.
[RIJ 12] also developed an automated stroke scoring system. This work describes an automated brain densitometry method to detect early ischemic changes by comparing contralateral density histograms. Also in this method, an atlas-based segmentation was performed. Histograms of 10 atlas regions were matched with its contralateral histograms. [RIJ 12] investigated the reliability of the proposed early ischemic changes detection method by comparing to manual detection by two experienced neuroradiologists. It was shown that the difference with manual scoring was in the same order of the interobserver variability. This study actually also showed that densitometry suffers from a sub-optimal interobserver agreement.

Because of the required registrations of the image data with atlases, these methods are quite computationally demanding. This is the reason why these methods have not been introduced yet into clinical practice.

### 3.1.4 Automatic thrombus detection

Decisions about the most adequate treatment for acute strokes must consider the total volume of the occluding thrombus. Thus the detection and volume measurement of intravascular thrombus is an important task. A few studies proposed some automated steps for detection and volume measurement of the thrombus.

[1] The Alberta Stroke Program Early CT Score (ASPECTS), a 10-point scale, is a clinical tool for assessment of early ischemic changes after stroke based on the location and extent of a visible stroke lesion.

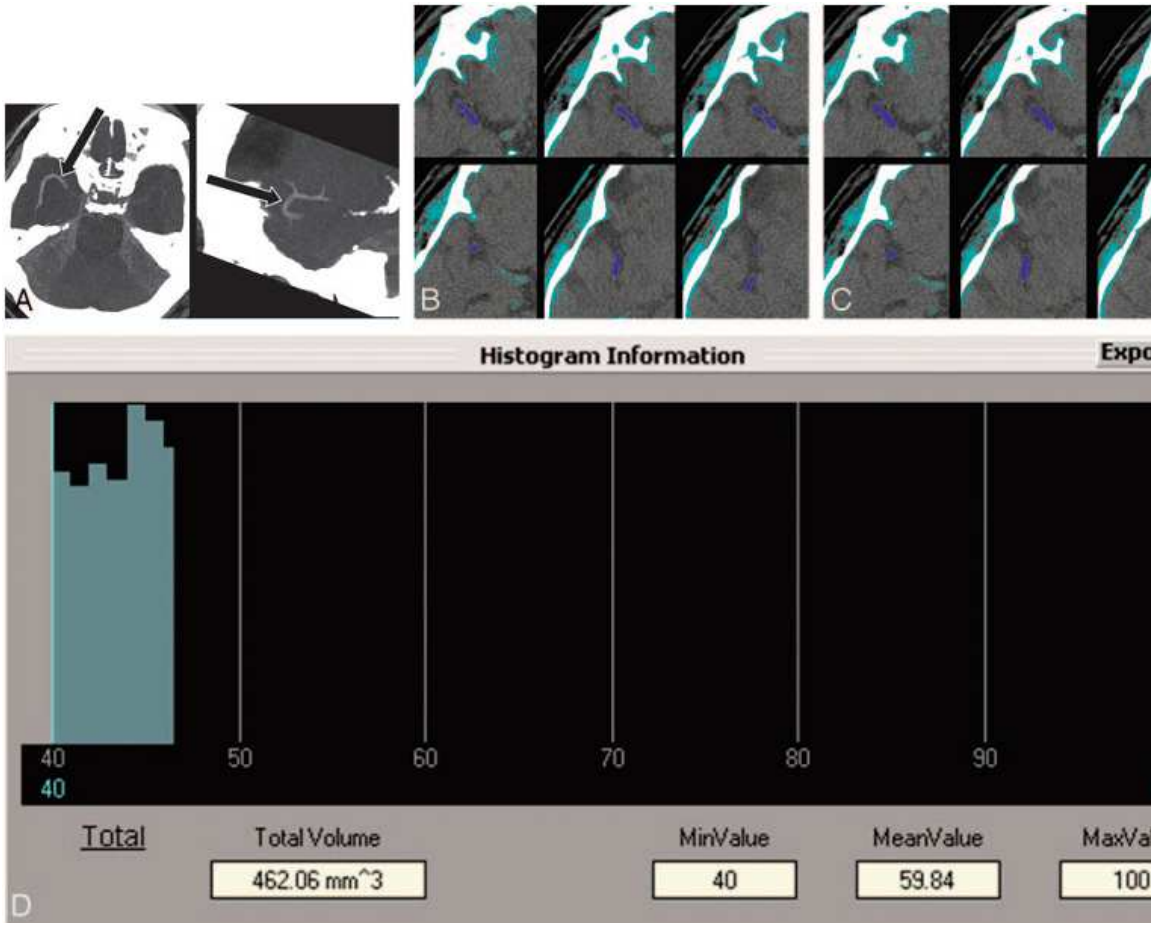


**Figure 12. Illustration of the thrombus volume measurement. A, maximum intensity projection images of non-contrast CT (arrows). B, pixels are segmented to determine the thrombus (azure areas). A region of interest is drawn around the thrombus. C, dilation with a single iteration of the segmented thrombus. D, calculation of the thrombus volume.**

In [KIM 08]a 3D region growing is performed after a manual thrombus selection. Next, the thrombus volume is defined by its resulting segmentation. Figure 12illustrates this process.
[RIE 10] proposes a semi-automated method for thrombus segmentation that calculates the volume and length of the segmented thrombus. This method starts with the manual delineation of a region of interest. Subsequently, a seeded region growing algorithm is performed. Thrombus volume is calculated as the sum of the segmented voxels volume. Thrombus length is defined as the length of thrombus medial axis. In Figure 13 this process is illustrated. This method requires a lot of user interaction and high quality images. As can be seen in Figure 13A, the area of the segmentation is very restricted. So the thrombus detection is basically a manual process. Also, the thrombus is clearly visible (Figure 13B), which is not very common in clinical practice.

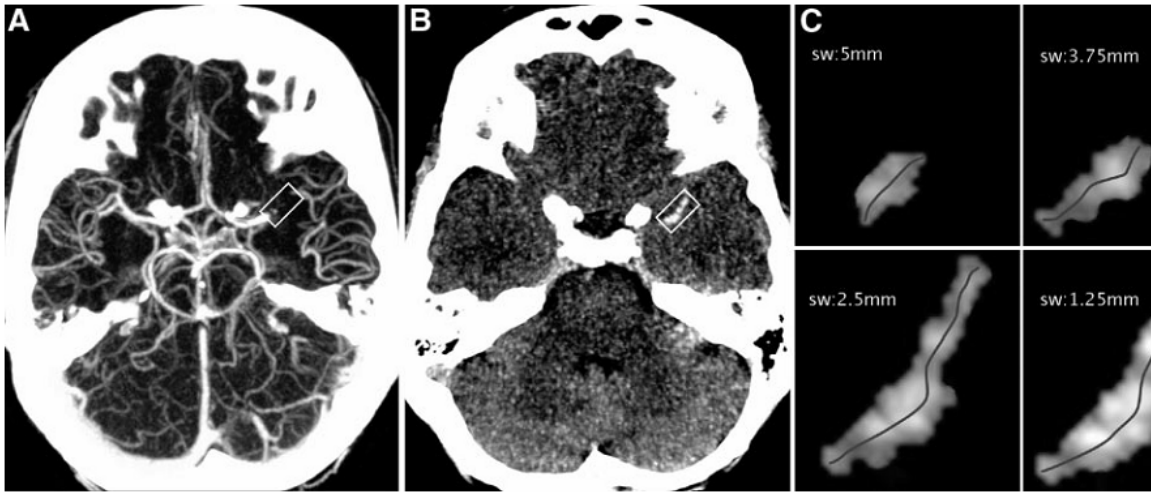


**Figure 13. A: maximum intensity projection (MIP) of a CT angiography (CTA) image from a patient with acute ischemic stroke marked inside the box on the left. B: an image showing a hyperdense middle cerebral artery sign in exactly the same location where the contrast gap is found in the CTA image. C: the 4 MIP images show the results of the segmentation of clot. The thin dark gray lines on the segmentation results represent the calculated middle axes of the clots.**

### 3.1.5 CT-Perfusion analysis

CT Brain Perfusion imaging (CTP) is emerging as a promising diagnostic tool for initial evaluation of acute ischemic stroke patients. With CTP images, areas with brain perfusion defects can be detected after the onset of clinical symptoms. CTP maps can facilitate the distinction between the irreversibly damaged infarct core and the potentially reversibly damaged infarct penumbra, which is important in choosing the most suitable therapy.

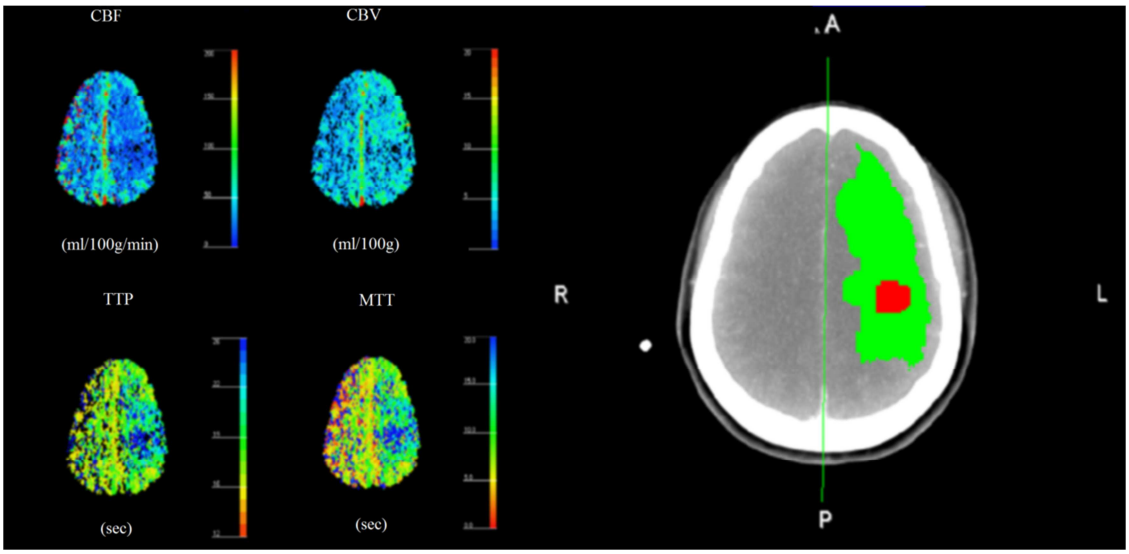


**Figure 14. Example of CTP analysis results: On the left CBF, CBV, TTP, and MTT maps are displayed; on the right the summary map is depicted: Red represents the infarct core area and green represents penumbra area.**

Perfusion information is obtained by monitoring the dynamic passage of iodinated contrast agent bolus through the cerebral vasculature and tissue. During CTP

acquisition, CTP source images are acquired every 1 to 3 seconds for approximately 1 minute. Time–attenuation curves of individual voxels are used to estimate local perfusion parameters such as cerebral blood volume (CBV), cerebral blood flow (CBF), mean transit time (MTT), and time to peak (TTP).

Post-processing steps are conducted to register time frame image volumes, segment the cerebral volume, and to define the arterial input, venous output, and cerebral midline. The arterial input function is required to perform a deconvolution with the time-intensity curves of the brain tissue. The venous output function is required to correct the arterial input for volume averaging effects.

Using pre-defined thresholds and contra-lateral comparison, these maps are combined to create a summary map estimating the volume of infarct core and penumbra. Wintermark et al. [WIN 09] proposed the selection of the ischemic area on basis of a relative MTT threshold, defined as the area in which the MTT is increased 1.5 times compared to the contralateral side. Commercial software packages for CTP analysis are widely available using various algorithms and perfusion models. As a matter of fact, there is currently no standardized method for the analysis. Other issues related to inter- and intra-user variability in defining parameters during pre and post processing, pitfalls / technical errors, and differences among vendors and software platforms, as well as misleading interpretation of suspected infarct area are now become big concern in CTP analysis.[GOY 13] [MAN 13] The necessarily of standardized and validated methods for CTP analysis drive the research in this area, including the development of the CTP phantom data for simulation. [RIO 11]

Dose and image quality are still key issues. Developing techniques for high-quality perfusion at acceptable dose levels are a main challenge. For example, Mendrik et al [MEN 11] proposed a 4D noise filter to improve image quality using information from the whole acquisition series of CTP data.

In years to come, the trend of the CTP research will be in the development of a combination of CT angiography (CTA) and CT perfusion as a one-stop shop for stroke imaging. Smit et al [SMI 13] have shown that a relatively simple technique, Timing-Invariant CTA derived from CTP, was able to show collateral vessels better than conventional CTA. TI-CTA utilizes 4-D data series of CTP and display vessels independently from arrival of contrast, while CTA only provides snapshot of the collaterals.

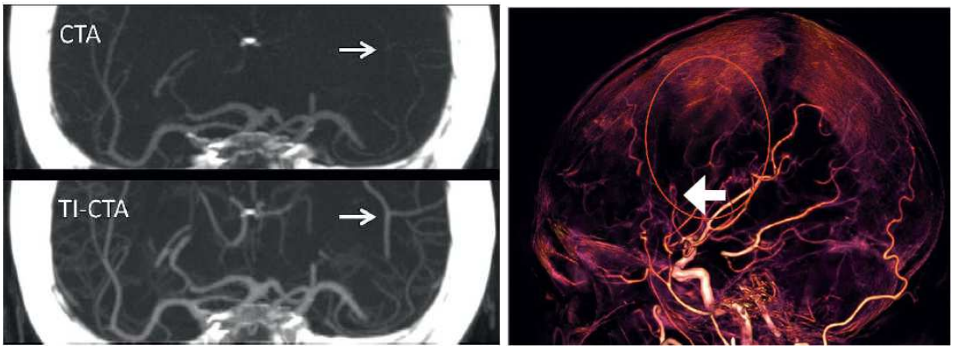


**Figure 15. Comparison of conventional CTA and TI-CT [SMI 13] on the left. On the right a high-resolution CTA derived from CTP data is shown.**

## 3.2 Image processing for (Multi-) trauma

In a recent outcome study [Gar 07] the integration of whole-body CT into early trauma care significantly increased the probability of survival in multiple trauma patients. The first reading takes place in the console room on thick axial images, while images are reconstructed. Original thin-slice image data are often accompanied by different bone and soft-tissue reconstructions, MPR reconstructions, maximum intensity projections or volume renderings. All thin- and thick-slice reconstruction results generated at the scanner need to be read and analyzed during a second reading in the reading room. Workflow enhancing navigation and viewing tools improves the efficiency of the radiologists' review process.

Medical disciplines other than radiology, such as trauma surgery, interventional radiology, and neurosurgery, have to view images without delay or find the crucial information in reports. Therefore post-processing applications have to provide tools that allow the radiologists to easily demonstrate the major pathologies.

### 3.2.1 State of art on Trauma Related Segmentation Techniques

Examination of a CT scan of a multi-trauma patient consists of many inspections including: vascular injuries, hemorrhages, injury of airways, spinal injuries, injury of head and brain, and skeletal injuries. Below these examinations and potential tooling are described.

#### *3.2.1.1 Vascular Injuries*

CT angiography (CTA) quickly identifies vascular injury, determines its severity, points to the source of bleeding, and guides treatment decisions. CTA is used to examine blood vessels in the following areas of the body:

- head and neck
- heart and mediastinum
- lungs
- abdomen (vital organs such as the spleen, kidneys and liver)
- pelvis
- extremities

Vascular injury is often detected by the presence of blood, contrast extravasation (blush), or a cut-off of an artery. Because of its high protein content, unclotted extravascular blood usually has a measured attenuation of 30-45 HU; however, its attenuation must be assessed on the basis of the individual patient. For example, hemoperitoneum may have a measured attenuation of less than 30 HU in a patient with a decreased serum hematocrit level or in a patient with a haemorrhage that is more than 48 hours old (the attenuation of clotted blood is 45-70 HU). The body attempts hemostasis by forming a blood clot at the site of injury. Thus, on CT images, the highest-attenuation hematoma, or sentinel clot, is that closest to the site of bleeding, whereas lower-attenuation unclotted blood is located farther from the source. An active bleeding (>100 HU) is typically surrounded by hematomas [Lub 07]. The length of the injury and the diameter of the vessel above and below the injury are important for e.g. stent selection.

#### 3.2.1.2 Head and neck vessel injuries

A non-contrast study of the brain is performed prior to the CTA examination of the head and neck. The percentage of vascular narrowing, particularly in the arteries of the head and neck, identifies the presence and location of hemorrhages (intracerebral, subdural, epidural subachnoidal) and the presence of mass effect, midline shift and swelling. Vessel inspection may benefit from algorithms that automatically detect the centerline through the vessel [Bul 05].

#### 3.2.1.3 Ruptured Abdominal Aortic Aneurysms

An Abdominal Aortic Aneurysm (AAA) is an abnormal enlargement of the abdominal aorta. Such an aneurysm may rupture leading to an acute medical emergency situation. Less than 80% of patients survive a ruptured abdominal aneurysm. The standard treatment of a ruptured aneurysm is open abdomen aneurysm repair. Alternatively, the ruptured aneurysm can be treated with an endovascular stent grafting. This may be a safer approach but is only suitable for patients with specific anatomical measures. Assessment of anatomical suitability is important for patient selection. Current AAA visualization and measuring tooling has been developed for elective patients only. It was recently concluded that current post-processing algorithms are sub-optimal for patient selection in such an emergency situation [Sch 08]. The lack of contrast enhancement in the distal arteries was the main cause of the hampering of an automated analysis.

#### 3.2.1.4 Spinal Injuries

Dependent on the type of trauma, the entire spinal column is inspected for osseous and soft tissue injuries. The centerline through the spinal cord is automatically detected [Ran 06], [Arc 02]. This centerline is used to create curved MPR's with that greatly improves inspection of the spinal column. The vertebrae must be labeled to facilitate reporting. Cross-sectional MPR's of both the vertebrae and the intervertebral discs positioned orthogonal to the centerline, facilitates the search for lesions.

#### 3.2.1.5 Skeletal Injuries

Damage Control Orthopedics should be regarded as a part of the resuscitation process. By careful choice of surgical technique, blood loss and tissue trauma can be minimized [Pap 02], [Hil 04]. It is well documented that 'unstable' pelvic fractures are associated with massive haemorrhage [Cry 88] and haemorrhage is the leading cause of death in patients with major pelvic fractures. Therefore, automated fast and accurate skeletal structures and contrast blush segmentation and visualization greatly improve skeletal injury assessment [Kan 03], [Zor 03].

### 3.2.2 Conclusions

- Hypodensity in CT images helps the detection of extravascular blood and vascular injuries.
- Automatic spinal cord detection and vertebrae labeling facilitates the search for presence of lesions in the spinal column.

- Highly automated quick and accurate skeletal structures segmentation and visualization greatly improve skeletal injury assessment.
- Image analysis of emergency ruptured abdominal aortic aneurysms is important for patient selection, but its robustness should be improved.

## 3.3 Image processing for Oncology

Cancer is one of the leading causes of death in the world and costs more in productivity and lost life than any other illness, according to a new report published by the American Cancer Society [Gar 07]. With the increase in the aging population and changes in lifestyle behavior, the incidence of cancer is on the rise globally (economically developed and developing countries). Imaging modalities (X-ray, CT, MR, PET, US) play a significant role in detection, diagnosis, staging and monitoring of cancer and also in image-guided treatments like RF ablation. Diagnostic oncology (tumor detection, diagnosis, and monitoring) related scanning account for a majority of all CT procedure volume. Introduction of new drugs and therapies also require continuous and frequent monitoring of the patient to analyze their influence and effect.

Post-processing of oncology (CT, PET/CT, and MR) image datasets can pose a major challenge for radiology and oncology staff. It often requires the maneuvering and analysis of a significant amount of data due to multiple follow-up studies. The user has to:

- Load multiple patient examinations for side-by-side comparisons to the baseline scan and nadir scan (smallest criteria recorded since the treatment started) in case of new progression. These datasets often include exams from multiple imaging modalities.
- Scroll through the numerous images of large datasets to identify and locate the tumors and lymph nodes for follow-up analysis.
- Measure the size of the tumors and lymph nodes including long axis and short axis diameters, as well as other quantitative parameters such as HU values.
- Calculate tumor burden measurements based on RECIST and WHO criteria and assess response to treatment.

Dedicated tools help simplify the review and analysis of multi-modality oncology datasets for tumor detection and monitoring. Semi-automatic segmentation tools facilitate 2D and 3D segmentation of tumors and lymph nodes. For instance, it allows the user to make quick bi-dimensional measurements using 2D measurements tools. Semi-automated measurements are more reproducible than manual measurements made by radiologists [Lub 07]. Reporting tools also enable graphical tracking of the size of the tumor across the different time points. On completion of the measurements, RECIST and WHO calculations (see Abbreviations and Definitions) are calculated automatically. For follow-up assessment, automatic registration of the current and previous datasets reduces the need to scroll through the datasets in order to locate the target tumors, and this improves workflow.

It is imperative that any innovation should seamlessly be integrated into the (streamlined) workflow to improve communication and reduce time to report. A Streamlined workflow for the evaluation of oncology cases is described below.

1. An oncologist requests a CT scan needed for initial detection, staging or follow-up analysis.
2. The obtained CT scan is assessed by the radiologist.
    a. For initial detection and staging
        i. Radiologist identifies the lesions.
        ii. Uses semi-automatic segmentation tools for tumor segmentation to obtain reproducible measurements.
        iii. Generates a report.
    b. For follow-up
        i. Automatic registration of CT datasets allows single-click identification of lesions in follow-up datasets.
        ii. Uses semi-automatic segmentation tools for tumor segmentation to obtain reproducible measurements [Lub 07].
        iii. Generates a report.
3. The report is send to the oncologist. The automatically calculated tumor burden based on RECIST/WHO criteria is added to the report.

The ability to access, create and communicate actionable information throughout the institution is essential. Collaboration tools enables radiologists to discuss a case and share information (e.g. tumor parameters, tumor burden, key images) with oncologists to improve patient care.

### 3.3.1 State-of-the-art on Tumor Segmentation Techniques

In oncology segmentation tools can be used to:

- Facilitate 2D and 3D segmentation of tumors and lymph nodes. As a result, it allows the user to make quick bi-dimensional measurements using 2D measurements tools. Semi-automated measurements are more reproducible than manual measurements made by radiologists [ZHA 09].
- Facilitate registration of e.g. the pulmonary region in CT follow-up studies where lungs are automatically delineated.

#### *3.3.1.1 Tumor segmentation in PET imaging*

$^{18}$F-FDG PET functional imaging is increasingly used in clinical and therapeutic care to complement CT[NES 08]. It provides a better detection and discrimination of tumoral tissues. For mediastinal lymph node evaluation of patients with poorly defined non-small cell lung cancer on CT-based treatment planning images, it has been shown that fusion with PET often leads the radiation oncologist to modify the definition of gross tumor volume (GTV) by excluding regions of atelectasis or pneumonitis [LAS 11]. Moreover, as $^{18}$F-FDG PET imaging is more sensitive and specific than CT for the detection of lymph node metastases and malignant tissues [PIE 00][DWA 99], the inclusion of $^{18}$F-FDG PET in planning may also lead to an increase in the size or number of GTVs. Actually, the use of $^{18}$F-FDG PET has shown its ability to modify GTV volumes and shapes compared to those obtained on CT images. It also decreases inter-observer variability in GTV delineation. So, determination of a volume of interest from $^{18}$F-FDG PET imaging is also crucial for radiotherapy treatment planning.

Several methods have been proposed in the literature to define the edge of $^{18}$F-FDG PET positive tissue. The most intuitive is a manual delineation by an experienced physician or by a radiation oncologist. This method is tedious, time consuming, with lack of reproducibility leading to a large source of errors [NES 07][NES 07]. Many methods have been proposed in the literature, some more sophisticated than others. They include region growing algorithms [GRE 08][GRE 08], deriving methods[DRE 07], parametric approaches[ARI 07], Markov Random Field theory [MON 07][HAT 07] and watershed algorithms[DRE 07][GEE 07]. The main approach proposed in the literature, however, is thresholding. The most commonly used is a constant threshold around 40% of the maximum uptake within the lesion ([ERD 97][NES 07]) or a fixed Standard Uptake Value (SUV), for example 2.5 [PAU 04][NES 07]. For volumes larger than 4 mL, [ERD 97]proposed a threshold value that ranged from 36% to 44%, depending on the theoretical source-to-background ratio.

The simple approaches are not valid for small and poorly contrasted positive tissue. For these cases, more complex thresholding algorithms have been proposed to define the optimal threshold value to be applied to segment the lesion. The principle is to adapt the threshold following a fitting model according to one or two characteristic parameters of the lesion. The relationship between the optimal threshold and the parameters is defined during a calibration procedure using tomographic phantoms. Various parameters have been proposed in the literature, such as lesion volume and contrast[ERD 97], the SUV [BLA 04] or the contrast[DAI 03][NES 07]. Some of them are true (ideal or experimental) [ERD 97], others are measured parameters ([DAI 03][BLA 04][NES 07]). When they are measured, the measurement is carried out on a single voxel, or corresponds to a mean value. For background, it can be a global measurement at a distance from the lesion ([DAI 03][JEN 07]) or a local one within or near the lesion [NES 07]. Some methods use an iterative approach allowing a sequential estimate of the optimal threshold without *a priori* knowledge from anatomic images ([JEN 07][JEN 07]). It is not straight-forward to objectively choose the best fitting model from all these approaches.
The purpose of Vauclin et al. [VAU 09] was to develop a generic thresholding algorithm, independent of the thresholding model used, and to apply it on 3 fitting models based on approaches proposed in the literature ([ERD 97][DAI 03][BLA 04]). Finally, Tylski et al. [TYL 10] proposed a new thresholding approach and brought an evaluation of accuracy and robustness compared to other methods on simulated, experimental, and clinical data.

#### *3.3.1.2 PET guided CT segmentation*

Radiotherapy planning is usually based on CT images, and requires the segmentation of pathologies as well as of organs at risk to carefully design the radiation region. To segment tumors or lymph nodes, the bi-modality imaging PET/CT is exploited, in which PET information is used to constrain the CT segmentation. This automatic approach is described in the following paragraph.

##### 3.3.1.2.1 Variational image segmentation

For lung tumors it is often difficult to separate the tumor from the surrounding vessels or mediastinal tissues in CT image data. In this case, PET images can provide useful information to guide the segmentation on the CT images, highlighting only tumoral tissues.
Variational image segmentation methods consist in finding contours or regions in a given image by minimizing an appropriate energy function to minimize weighted sum

between a CT based energy and a PET based energy. [WOJ 10] Figure 18 shows an example of this approach.

#### *3.3.1.3 Dedicated tools for contouring in images from multiple modalities*

Some dedicated radiotherapy software tools to aid contouring exist. They allow the radiologist to trace the contours such that this information can be included in the radiation planning to spare relevant organs at risk during the irradiation treatment. Two approaches based on the CT images are currently available.

First, atlas based segmentation (ABAS) uses atlases, which are pre-existing contoured 3D CT images, to make contours of patient anatomical structure boundaries. ABAS registers these atlases with the patient CT series such that the atlas contours are transformed onto the patient specific CT images. These structures are subsequently modified by the treating physician. Known solutions are presented by DOSIsoft SA (France), Elekta AB (Sweden), and Velocity Medical Solutions (USA) provide atlases for head and neck and/or pelvis localizations.

Second, many semi-automatic segmentation methods of the healthy organs have been implemented for the contouring of the Treatment Planning System (TPS) in radiotherapy. Dedicated tools have been developed by the manufacturers of TPS software for the segmentation of bones, lungs, breasts and spine. However, due to low contrast on CT, many organs are much more difficult to segment, e.g. liver and bladder. In the case of tumor segmentation, it is desired to take advantage of contours made from automatic segmentation methods on anatomical modalities (CT, MRI) and functional modalities (PET, SPECT) in proposing these contours to the user within a selection and shaping tool.

#### *3.3.1.4 Tumor segmentation tools for PET/CT images*

DOSIsoft has developed a multi-modality imaging and diagnostic workstation called PLANET Onco. It allows the user to manipulate hybrid PT/CT images for segmentation, quantification and monitoring tumor changes in PET FDG to follow-up the therapeutic process of treatment and to assess its effectiveness. This software is successful and innovative, as it incorporates methods to segment tumors as well as tools for quantitative monitoring of tumor changes.
Semi-automatic thresholding based segmentation methods have been implemented in PLANET Onco. The tumor boundaries are determined automatically after a rough manual definition of a volume of interest around the lesion.. This allows the measurement of quantitative values associated with each thresholded lesion such as volume and tumor activity (maximum SUV, peak SUV, mean SUV).
These automatic approaches are described below.

##### 3.3.1.4.1 Maximum Intensity

This is a thresholding method, which is the most commonly used method. It works with a constant threshold around 40% of the maximum uptake within the lesion or a fixed SUV.[ERD 97], [PAU 04], [NES 07]

##### 3.3.1.4.2 Black method

The Black method is based on a linear model. [BLA 04] The optimal threshold $Th_{opt}$ (expressed in SUV) is defined using:

$$Th_{opt} = Mean_{int} \times A_1 + A_2.$$

Where $Mean_{int}$ is the mean intensity in the thresholded volume of interest expressed in SUV. $A_1$ and $A_2$ are constant parameters computed using a phantom, both values are dependent of the used camera and images reconstruction parameters. A first threshold is initially set to 40% of the maximum intensity. $Mean_{int}$ is subsequently computed within this volume. This value is used to determine a new threshold value using the equation above. A new volume is subsequently defined resulting to a new $Mean_{int}$. This procedure is repeated until the threshold value convergences.

#### 3.3.1.4.3 Nestle method

The Nestle method [NES 05][NES 07]is also based on a linear model with $Th_{opt}$ defined as follow:

$$Th_{opt} = Mean_{int} \times B + B_{avg}$$

$Mean_{int}$ is here the mean intensity within the volume defined at 70% of the maximum intensity. $B_{avg}$ is the mean measured intensity of the lesion background. $B_{avg}$ is estimated from a shell region of 8 mm thickness surrounding the included tissue. The inner edge was chosen to be 8 mm away from the lesion boundary to limit the influence of PVE. $B$ is a constant parameter computed using a phantom and is dependent of the used camera and reconstruction parameters. This algorithm does not need iterations; it directly gives the optimal threshold. Figure 16 shows an example of a lung tumor segmentation using Max intensity, Black and Nestle methods.

#### 3.3.1.4.4 Fitting method

The Fitting method [TYL 10] is based on an iterative algorithm that simultaneously estimates the mean intensities of the lesion and it local background and also takes into account the spatial resolution of the camera using the FWHM of the dispersion function.
The original image is first oversampled and thresholded using the Nestle et al. method. A rough model of the tumor (binary mask) is then determined. It is finally reduced by erosion, filtered using a Gaussian filter corresponding to the machine FWHM and sub-sampled. At each iteration, the modified model is compared to the reference VOI to estimate a similarity criterion in the sense of least squares between the model of the tumor and the original VOI. The calculation is iterated until the criterion of similarity continues to decrease.

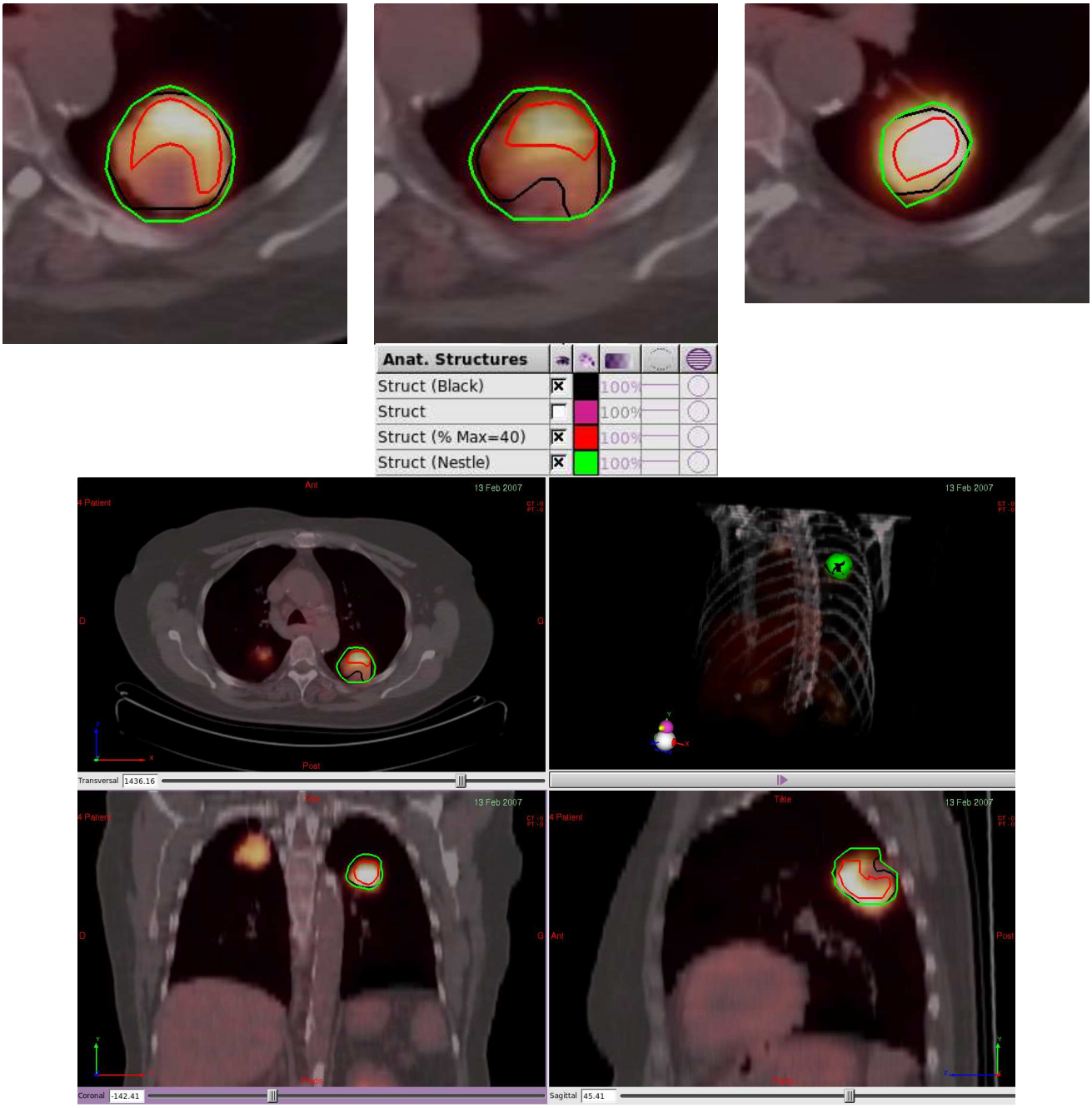


**Figure 16. Semi-automatic segmentation methods (Max intensity, Black, Nestle)**

#### 3.3.1.4.5 Adaptive

The adaptive method [VAU 09] is based on nonlinear modeling of $Th_{opt}$ with:

$$Th_{opt} = C_1 / Cont_{meas} + C_2$$

$Cont_{meas}$: the locally measured contrast. It represents the ratio between the averaged value of the maximum intensity in the VOI and $B_{avg}$ computed as described previously. $C_1$ and $C_2$ are constants computed using a phantom; both values are dependent of the used camera and reconstruction parameters. This algorithm estimates $Th_{opt}$ using an iterative procedure similar to the Black method. Figure 17 shows an example of a lung tumor segmentation using Fitting and Adaptive methods.

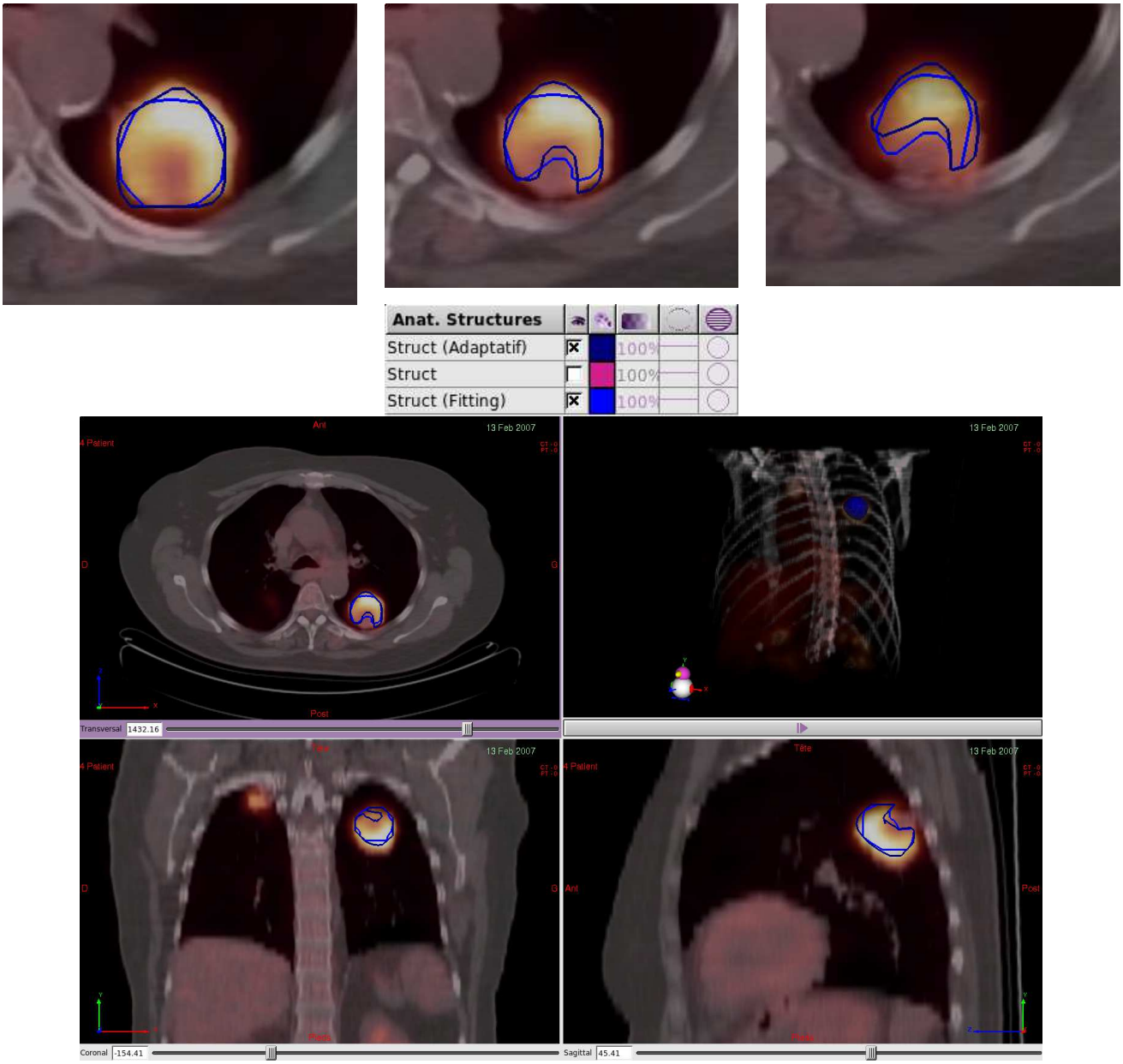


**Figure 17. Semi-automatic segmentation methods (Fitting, Adaptive)**

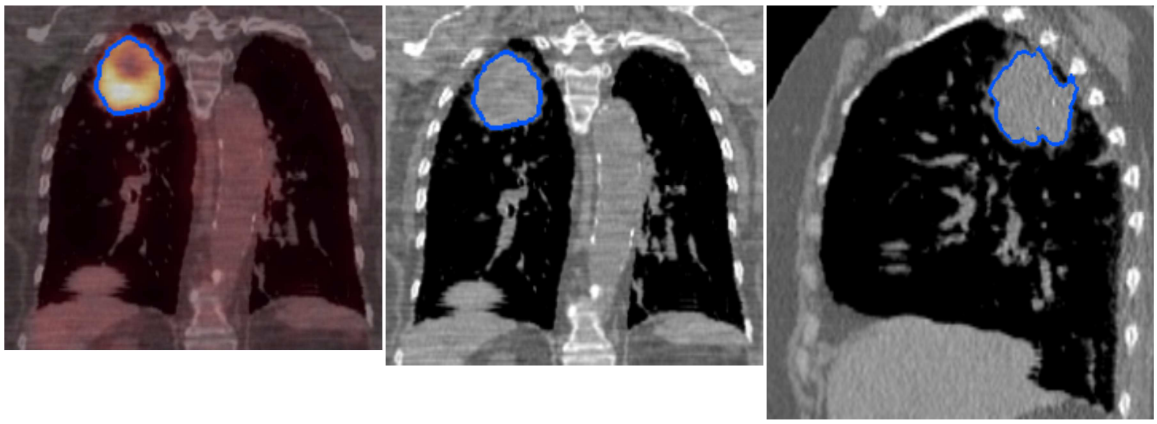

**Figure 18. CT Segmentation guided by PET**

#### *3.3.1.5 CT lung segmentation for registration*

A literature study on segmentation techniques that support pulmonary region delineation in CT follow-up studies resulted in a number of methods that all use the CT scans provided by the Lobe and Lung Analysis 2011 (LOLA11) challenge. Some methods use simple region growing algorithms combined with morphological operations, others combine these simple methods with more advanced and elaborated functions like e.g. atlas based segmentation techniques as fallback scenario if the latter fails. Some of the articles are briefly touched upon below.

The article “Automatic lung segmentation from thoracic computed tomography scans using a hybrid approach with error detection” [RIK 09] describes a method in which the lung segmentation applied consists of three steps. As a first step, the lungs are automatically segmented using a conventional method employing region growing and morphological smoothing. Next, automatic error detection is applied. The scans that are likely to contain errors are then segmented by a multi-atlas segmentation.

The article “Lung and Lung Lobe Segmentation Methods at Fraunhofer MEVIS” [LAS 11]describes a method that starts with a low-resolution (voxel size 2x2x3mm) analysis of the CT data to first locate and then coarsely segment the pulmonary airspace with a fixed-threshold region growing. Shape analysis, watershed segmentation, and morphological operations finally result in the requested pulmonary regions.

The article “Robust Active Shape Model Based Lung Segmentation in CT Scans” [SUN 11] describes a method which is based on an active shape model for lungs in combination with a robust model matching method. The statistical lung model guides and constrains the segmentation process, and thus successfully segments pathological lung tissue which can have similar density to tissue adjacent to the lungs.

### 3.3.2 CT Lung Nodule segmentation

A literature study on automatic segmentation techniques that detect the lung nodules in CT datasets results in a number of methods that all use the CT scans provide by the ‘Nelson study’, the largest CT lung cancer screening trial in Europe. The final outcome of this research is published in "Comparing and combining algorithms for computer-aided detection of pulmonary nodules in computed tomography scans: the ANODE09 study". Combining the findings of different systems appears to be a very powerful method to improve the performance of CAD systems. The combination of six CAD algorithms is able to detect 80% of all nodules at the expense of only two false

positive detections per scan and 65% of all nodules with only 0.5 false positives. This suggests that blending detection algorithms is a promising direction for future research in CAD.

### 3.3.3 State-of-the-Art on Volume Registration Techniques

Oncology scans require frequent comparison with previous datasets, to identify tumors and make measurements. Fully automatic registration of the current and previous datasets of the same modality (e.g. CT-CT) or registration of datasets of different modalities (e.g. PET-CT) simplifies assessment of follow-up exams.
There are roughly two types of registration techniques. For rigid structures rigid registration techniques suffice. For non-rigid structures e.g. lungs, liver rigid as well as non-rigid registration techniques may be used. The registration result of rigid registration techniques applied to non-rigid structures highly depends on the amount of deformation present between the different structures. A literature study on registration for oncology follow-up studies resulted in a number of papers on fully automatic registration techniques tuned to the specific clinical cases. Some of the articles are briefly touched upon below.

#### *3.3.3.1 Rigid Registration Techniques*

3.3.3.1.1 CT Lung registration

Article “Fast volumetric registration method for tumor follow-up in pulmonary CT exams” [SIL 11] contains a comprehensive overview on the latest used methods to register CT lung follow-up datasets. This overview contains rigid as well as non-rigid registration techniques. The preference for rigid registration over non-rigid registration for pulmonary CT exams and tumor follow-up is substantiated with the argument that the main objective of the transformation is to globally align the two pulmonary exams, but this alignment should not modify significantly the shape and volume of the tumor [BLF 04].
The proposed non-rigid registration methods perform the segmentation of the lungs and build a 3D image of the pulmonary region. Next, the center of mass is computed and the exams are coarsely aligned. Then the 3D registration is performed using a downsized volume of the original CT exam. Mutual information[2] is the most discussed and acclaimed technique that is successfully used as a comparison function for registering datasets [VIO 95]. The outcome of the comparison function is fed into an affine or a rigid body transformation, which is used to actually align the different datasets. An affine transformation[3] is slightly better than a rigid body transformation[4]. The affine transformation on a previously segmented lung volume, which in total is not much slower than the transformation without segmentation, shows better alignment and higher robustness[BLF 04].

3.3.3.1.2 PET / CT Liver registration

The article “Evaluation of Image Registration in PET/CT of the Liver and Recommendations for Optimized Imaging” [VOG 07]describes that hybrid PET/CT of the liver may have significant registration errors and artifacts related to breathing motion. The extent of these issues depends on the selected breathing protocol and the speed of the CT scanner. No protocol or scanner can guarantee perfect image

[2] Mutual information is a measure of how much information one image contains about another.
[3] Affine transforms allow rotation, translation, scaling, and skew.
[4] Rigid body transforms allow only rotation and translation.

fusion. On the basis of these findings, recommendations were formulated with regard to scanner requirements, breathing protocols, and reporting.

The article “Accuracy of rigid CT-FDG-PET image registration of the liver” [DAL 04] describes a semi-automatic, organ-focused method to minimize the uncertainty well beyond the typical uncertainty of 5-10 mm obtained by commonly available methods. By restricting registration to the liver region and by isolating the liver on computed tomography (CT) from surrounding structures using a threshold technique, registration was achieved using the mutual information-based method. Unlike the commonly available methods (a manual, a landmark-based and a 'standard' mutual information-based method), this approach allows for robust CT-FDG-PET registration of the liver, with an accuracy better than the spatial resolution of the PET scanner that was used.

#### *3.3.3.2 Non-Rigid Registration Techniques*

Non-rigid registration remains a very active research topic.

##### 3.3.3.2.1 CT Lung registration

The article “Evaluation of Registration Methods on Thoracic CT: The EMPIRE10 Challenge” [MUR 11] describes a comprehensive evaluation and comparison of 20 individual algorithms from leading academic and industrial research groups.
All algorithms are applied to the same set of 30 thoracic CT pairs. Algorithm settings and parameters are chosen by researchers’ expert in the configuration of their own method and the evaluation is independent, using the same criteria for all participants. Many algorithms performed extremely well on many scan pairs. Overall was concluded that the use of lung masks is to be recommended for optimal performance in registration of the lung volumes. Regarding algorithm speed it is difficult to make generalizations since participants used own hardware, which varied greatly and since not all algorithms are designed and programmed for optimal efficiency.

##### 3.3.3.2.2 CT Liver registration

The article “Liver registration for the follow-up of hepatic tumors” [CHA 05] describes an efficient tree matching method applied on different segmentations of the vascular system of a single patient. These vascular systems are segmented from CT-scan images acquired (every six months) during disease treatment, and then modeled as trees. The method matches common bifurcations and vessels. Secondly, an estimation of liver deformation is computed from the results of the first step. This approach is validated on a large synthetic database containing cases with various deformation and segmentation problems. In each case, after the registration process, the liver recovery is very accurate (around 95%) and the mean localization error for 3D landmarks in liver is small (around 4mm).

### 3.3.4 Conclusions

- Blending detection algorithms is a promising direction for future research in CAD of pulmonary nodules in CT datasets.
- Rigid registration techniques are state-of-the-art in modern practice while non-rigid registration techniques remain a very active research topic.
- Rigid registration techniques can be used to register flexible structures in follow-up datasets but the outcome of the registration highly depends on the nature of the difference between the follow-up datasets.
- Rigid registration techniques are most commonly used to register CT follow-up datasets.

- Rigid registration techniques may use a previously generated segmentation result to become more robust (e.g. segmentation of the pulmonary region).
- Rigid registration techniques use mutual information as a comparison function for registering datasets.
- Rigid registration techniques use affine transformation because these perform slightly better than rigid body transformation.

## 3.4 Image processing in the cell-based in-vitro diagnostics

### 3.4.1 Introduction

Imstar currently produces the high-content slide scanning system PathFinder, which serves as the common platform for all its in-vitro diagnostic applications. This system constitutes the starting point of our work in the Medusa project.

### 3.4.2 Application workflow

The system workflow includes the following phases:

#### *3.4.2.1 Identification of the sample and support*

During this phase the user supplies the ID of the slide and the information concerning the sample and conditions of its preparation. Parts, sometimes significant, of this information can be inserted in the DB outside of the main PathFinder application, using the other applications or SQL scripts. The identification of slides being loaded into the acquisition hardware can be performed by the platform automatically, using the bar-coded slides and the integrated barcode reader. Further, PathFinder platform comprises slide zone ID management, supporting welled slides and other structured preparations, such as TMAs (tissue micro-arrays).

#### *3.4.2.2 Acquisition of the images*

Image acquisition can be performed either manually or automatically; the second option requires the appropriate hardware. Automated acquisition is accomplished via user-given acquisition protocol specification, which may include focus movement, filter/illumination changes, XY stage movement and time lapse, in any combination. The automated slide loader can be used, allowing unattended scans of multiple (up to 200 slides) slides. Focus movements allow acquisition of the specimens thicker than the objective focal plane and parfocality corrections between different illumination conditions. Filter/illumination changes allow capturing signals from multiple fluorescence dyes or multiple colors. XY stage movements can be performed such that any part of the slide surface is scanned or to ensure that the images are taken only at specific positions. Time lapse acquisition makes it possible to follow a process development in time.

#### *3.4.2.3 Detection of the objects of interest*

Once the images are acquired, the PathFinder proceeds with the detection of the objects of interest, such as cells, nuclei, signals from fluorescent labels etc. The detection procedure follows an objects model that is developed individually for every objects type and preparation. The model (i.e. objects and noise description) is external to the application code so it can be modified without changing the application. Further, as it is stored in text files, the model can be adjusted without implication of Imstar personnel. All the object features listed below (in the Characterization section) can be used in the model. The following procedures are available for the use in the detection procedures:

1. Image arithmetic: a formula in the form F(src1,src2)=dst is evaluated for every corresponding pixel pair of the source images to form the resulting image.
2. Affine transforms: rotation, translation, scaling, their combination and, in general, any linear transform determined by 2x3 matrix.
3. Linear neighborhood operations (convolution): given the 2D value matrix (kernel), we compute the destination pixel value as the weighted according to the kernel sum of the source pixel neighborhood. In the case of significant kernel size (more than ~10 pixels) the convolutions are performed in the frequency domain, using Fast Fourier Transform.
4. Non-linear neighborhood operations (mathematical morphology operations such as filling/contouring, rank, dilation/erosion, opening/closing, thinning/thickening, skeletisation/Voronoy tessellation, etc.).
5. Tree (grid) processing: Dijkstra path, nodes/branches filtering etc.
6. Watershed building and filtering: this technique consists of building the watershed grid for an imaginary relief where the height of every point is proportional to the corresponding pixel value. In practice, image smoothing is necessary prior to building the watershed, because otherwise we obtain too many tiny 'lakes' corresponding to insignificant local minima. The watershed separates every object in many parts; most of these parts are linked back together using the ratio of 'lakes' depth to the 'ridge' height parameter, that should not be too high.
7. Threshold-based image binarisation (given a source image and two threshold values, generate a binary image where the pixel value of 1 corresponds to the source pixel value between the thresholds and 0 to all other pixels). From one side, this is just a special case of the image arithmetic; however, it merits separate consideration because of its role in the segmentation. There exist multiple algorithms for the threshold values optimization; they are usually based on the analysis of the pixel value distribution of the source image, and/or the local contrast, and/or the morphological properties of the resulting binary image.
8. Labeling the connected components of the binary images and filtering them.

#### *3.4.2.4 Characterization and classification of the detected objects*

Once the objects are detected, PathFinder computes a list of characteristics (features) for each object. The feature list is defined in a text file external to the application, just like the object model description files. The list of features includes parameters of the following types:

1. Image: characterize the whole image (width, height, integral intensity etc).
2. Identification: name of the image, labels and marks found over it, ROIs it lays within etc.
3. Morphology: object's properties such as surface, perimeter, length, width, orientation, distance function measurements etc.
4. Counting: evaluate parameters of the second binary mask (grains/spots/dots) relative to the first one (cells/nuclei/fibres/chromosomes). For simplicity, these masks are referenced below as grains and cells, respectively.
5. Positioning within the parent: can be used only in spots results table. These parameters define position of an individual spot relative to the parent object's axis.

6. Intensity/Optical density: characterize the distribution of the pixel values inside the object: moments- and texture-related parameters can be evaluated (mean/mode/median).
7. Background: the background (outside of all the detected objects) is characterized separately, as a distinct object.

#### *3.4.2.5 Results analysis and presentation*

The table of results can be presented in various forms: tabular, graphic (plots or histograms) and as the objects gallery. In all the moments the application maintains the link between numerical or graphical results representation and the source images, corresponding to the objects being analyzed. Results can be presented for all the objects corresponding to the whole image set at once. Results tables can be exported for further processing (or presentations) by other applications (spreadsheets, statistical packages, databases etc.).

#### *3.4.2.6 Visual validation of the cell detection and classification*

The final step of the in-vitro diagnostic process is review and validation by the user (pathologist) of the cells classified as normal or alert, and drawing the conclusion (diagnostic) based on the validation.

## 3.5 References

[ARI 07] Aristophanous M., Penney B. C., Martel M. K. and Pelizzari C. A. A Gaussian mixture model for definition of lung tumor volumes in positron emission tomography. *Med Phys*, 34(11)4223-35, Nov 2007.

[BLA 04] Black Q. C., Grills I. S., Kestin L. L., Wong C.-Y. O., Wong J. W., Martinez A. A., and Yan D. Defining a radiotherapy target with positron emission tomography. *Int J Radiat Oncol Biol Phys*, 60(4) :1272-1282, Nov 2004.

[BLF 04] Thomas Blaffert and Rafael Wiemker. "Comparison of different follow-up lung registration methods with and without segmentation", Proc. SPIE 5370, Medical Imaging 2004: Image Processing, 1701 (May 12, 2004); doi:10.1117/12.535345.

[CHA 05] Arnaud Charnoz, Vincent Agnus et al. “Liver registration for the follow-up of hepatic tumors”, MICCAI 2005, LNCS 3750, pp. 155-162, 2005.

[DAI 03] Daisne J. F., Sibomana M., Bol A., Doumont T., Lonneux M., and Grégoire V. Tri-dimensional automatic segmentation of PET volumes based on measured source-to-background ratios : influence of reconstruction algorithms *Radiotherapy and Oncology***,** 69 247-50, Dec 2003.

[DAL 04] J.A. van Dalen, W. Vogel et al. “Accuracy of rigid CT-FDG-PET image registration of the liver”, Phys Med Biol. 2004; 49:5393–5405.

[DRE 07] Drever L. A., Roa W., McEwan A., Robinson D. Iterative threshold segmentation for PET target volume delineation. *Med Phys***,** 34 1253-65, Apr 2007.

[DWA 99] Dwamena B. A., Sonnad S. S., Angobaldo J. O., and Wahl R. L. Metastases from non-small cell lung cancer: Mediastinal staging in the 1990s—meta-analytic comparison of PET and CT. *Radiology*, 213 530 –36, Nov 1999.

[ERD 97] Erdi Y. E., Mawlawi O., Larson S. M., Imbriaco M., Yeung H., Finn R., and Humm J. L. Segmentation of lung lesion volume by adaptative positron emission tomography image thresholding. *Cancer* 80, (Suppl) 2505-09, Dec 1997.

[GAM 01] Gambhir S. S., Czernin J., Schwimmer J., Silverman D. H., Coleman R. E. and Phelps M. E. A tabulated summary of the FDG PET literature *J Nucl Med*, 42 1S-93S, May 2001.

[GEE 07] Geets X., Lee J. A., Bol A., Lonneux M., Grégoire V. A gradient-based method for segmenting FDG-PET images: methodology and validation. *Eur J Nucl Med Mol Imaging*, 34 1427-38, Sep 2007.

[GIN 09] B. van Ginneken, S.G. Armato et al. "Comparing and combining algorithms for computer-aided detection of pulmonary nodules in computed tomography scans: the ANODE09 study", Medical Image Analysis 2010;14:707-722

[GON 13] González, R. G., Copen, W. a, Schaefer, P. W., Lev, M. H., Pomerantz, S. R., Rapalino, O., Chen, J. W., et al. (2013). The Massachusetts General Hospital acute stroke imaging algorithm: an experience and evidence based approach. *Journal of neurointerventional surgery*, *5 Suppl 1*(figure 2), i7–12. doi:10.1136/neurintsurg-2013-010715

[GOY 13] Goyal, M., Menon, B. K., & Derdeyn, C. P. (2013). Perfusion imaging in acute ischemic stroke: let us improve the science before changing clinical practice. Radiology, 266(1), 16–21. doi:10.1148/radiol.12112134

[GRE 08] Green A., Francis R., Baig S. and Begent R. Semiautomatic volume of interest drawing for $^{18}$F-FDG image analysis-method and preliminary results. *Eur J Nucl Med Mol Imaging*, 35(2) :393-406, Feb 2008.

[HAT 07] Hatt M., Lamare F., Boussion N., Tuzro A., Collet C., Salzenstein F. *et al.* Fuzzy hidden Markov chains segmentation for volume determination and quantitation in PET. *Phys Med Biol***,** 52 3467-91, Jun 2007.

[JEN 07] Jentzen W., Freudenberg L., Eising E. G., Heinze M., Brandau W., Bockisch A. Segmentation of PET volumes by iterative image thresholding. *J Nucl Med,* 48 108-14, Jan 2007.

[KIM 08] Kim, E. Y., Yoo, E., Choi, H. Y., Lee, J. W., & Heo, J. H. (2008). Thrombus volume comparison between patients with and without hyperattenuated artery sign on CT. AJNR. American journal of neuroradiology, 29(2), 359–62. doi:10.3174/ajnr.A0800

[KOS 10] Kosior, R. K., Lauzon, M. L., Steffenhagen, N., Kosior, J. C., Demchuk, A., & Frayne, R. (2010). Atlas-based topographical scoring for magnetic resonance imaging of acute stroke. *Stroke; a journal of cerebral circulation*, *41*(3), 455–60. doi:10.1161/STROKEAHA.109.567289

[LAS 11] Bianca Lassen, Jan-Martin Kuhnigk et al. “Lung and Lung Lobe Segmentation Methods at Fraunhofer MEVIS”, LOLA11, 2011.

[MAH 02] Mah K., Caldwell C. B., Ung Y. C., Danjoux C. E., Balogh J. M., Ganguli S. N. *et al.* The impact of (18)FDG-PET on target and critical organs in CT-based treatment planning of patients with poorly defined non-small-cell lung carcinoma: a prospective study. *Int J Radiation Onc Biol Phys* 52 339-50, Feb 2002.

[MAL 01] Maldjian, J. a, Chalela, J., Kasner, S. E., Liebeskind, D., & Detre, J. a. (2001). Automated CT segmentation and analysis for acute middle cerebral artery stroke. AJNR. American journal of neuroradiology, 22(6), 1050–5

[MAN 13] Mangla, R., Ekhom, S., Jahromi, B. S., Almast, J., Mangla, M., & Westesson, P.-L. (2013). CT perfusion in acute stroke: Know the mimics, potential pitfalls, artifacts, and technical errors. Emergency radiology, (Table 1). doi:10.1007/s10140-013-1125-9

[MEN 11] Mendrik, A. M., Vonken, E., Van Ginneken, B., De Jong, H. W., Riordan, A., Van Seeters, T., Smit, E. J., et al. (2011). TIPS bilateral noise reduction in 4D CT perfusion scans produces high-quality cerebral blood flow maps.

Physics in medicine and biology, 56(13), 3857–72. doi:10.1088/0031-9155/56/13/008

[MON 07] Montgomery D. W. G., Amira A., Zaidi H. Fully automated segmentation of oncological PET volumes using a combined multiscale and statistical model. *Med Phys,* 34:722-36, Feb 2007.

[MUR 11] Keelin Murphy, Bram van Ginneken et al. “Evaluation of Registration Methods on Thoracic CT: The EMPIRE10 Challenge”, IEEE TRANSACTIONS ON MEDICAL IMAGING, 2011.

[NES 05] Nestle U., Kremp S., Schaefer-Schuler A., Sebastian-Welsch C., Hellwig D., Rübe C., and Kirsch C.-M. Comparison of different methods for delineation of 18F-FDG PET-positive tissue for target volume definition in radiotherapy of patients with non-Small cell lung cancer. *J Nucl Med*, 46(8) :1342-1348, Aug 2005.

[NES 07] Nestle U., Schaefer-Schuler A., Kremp S., Groeschel A., Hellwig D., Rübe C., and Kirsch C.-M. Target volume definition for 18F-FDG PET-positive lymph nodes in radiotherapy of patients with non-small cell lung cancer. *Eur J Nucl Med Mol Imaging*, 34(4) :453 462, Apr 2007.

[NES 08] Nestle U., Weber W., Hentschel M. and Grosu A. L. Biological imaging in radiotherapy: role of positron emission tomography. *Phys Med Biol,* 54 R1-R25, Dec 2008.

[PAU 04] Paulino A. C. and Johnstone P. A. FDG-PET in radiotherapy treatment planning: Pandora’s box? *Int J Radiation Onc Biol Phys,* 64 435-48, May 2004.

[PIE 00] Pieterman R. M., van Putten J. W. G., Meuzelaar J. J.*, et al.* Preoperative staging of non-small-cell lung cancer with positron emission tomography. *N Engl J Med,* 343 254–61, Jul 2000.

[REI 13] J.J. Reimerink, H.A. Marquering, A.C. Vahl et al. Semi-automatic sizing software in emergency endovascular repair for ruptured abdominal aortic aneurysms. Submitted, 2013

[RIE 10] Riedel, C. H., Jensen, U., Rohr, A., Tietke, M., Alfke, K., Ulmer, S., & Jansen, O. (2010). Assessment of thrombus in acute middle cerebral artery occlusion using thin-slice nonenhanced Computed Tomography reconstructions. Stroke; a journal of cerebral circulation, 41(8), 1659–64. doi:10.1161/STROKEAHA.110.580662

[RIO 11] Riordan, A., Prokop, M., Viergever, M., Dankbaar, J., Smit, E., & De Jong, H. (2011). Validation of ct brain perfusion methods using a realistic dynamic head phantom. Med Phys., 38, 3212–3221.

[RIJ 12] F. van Rijn, H.A. Marquering, M Staling, C.B. Majoie, B.C. Stoel, 2012, “Stroke scoring: Detection of early ischemic changes on non-contrast CT scans using an atlas-based segmentation of the ASPECTS areas. University of Twente, p: 72.

[RIK 09] E. M. van Rikxoort, B. J. de Hoop, M. A. Viergever, M. Prokop, and B. van Ginneken, “Automatic lung segmentation from thoracic computed tomography scans using a hybrid approach with error detection”, Medical Physics, vol. 36, no. 7, pp. 2934–2947, 2009.

[SHA 12] Sharma, B. (2012). Automatic Segmentation of Brain CT Scan Image to Identify Hemorrhages, 40(10), 1–4.

[SHI 12] Shieh, Y., & Chang, C. H. (2012). A utomated ASPECTS Scoring System as a Clinical S upport System for Acute Stroke Care, 25(Bhi), 691–694.

[SIL 11] José Silvestre Silva, et al. “Fast volumetric registration method for tumor follow-up in pulmonary CT exams”, Journal of Applied Clinical Medical Physics, Vol. 12, No. 2, Spring 2011

[SMI 13] Smit, E. J., Vonken, E. -j., Van Seeters, T., Dankbaar, J. W., Van der Schaaf, I. C., Kappelle, L. J., Van Ginneken, B., et al. (2013). Timing-Invariant Imaging of Collateral Vessels in Acute Ischemic Stroke. Stroke. doi:10.1161/STROKEAHA.111.000675

[SUN 11] Shanhui Sun, Christian Bauer et al. “Robust Active Shape Model Based Lung Segmentation in CT Scans”, LOLA11, 2011.

[TYL 10] Tylski P., Stute S., Grotus N., Doyeux K., Hapdey S., Gardin I., Vanderlinden B., and Buvat I. Comparative assessment of methods for estimating tumor volume and Standardized Uptake Value in (18)F-FDG PET. *J Nucl Med*, 51 : 268-276, Jan 2010.

[VAU 09] Vauclin S., Doyeux K., Hapdey S., Edet-Sanson A., Vera P., and Gardin I. Development of a generic thresholding algorithm for the delineation of $^{18}$FDG-PET-positive tissue: application to the comparison of three thresholding models. *Phys Med Biol,* 54 :6901-6916, Oct 2009.

[VIO 95] P. Viola and W. M. Wells III, “Alignment by maximization of mutual information”, International Conference on Computer Vision, E. Grimson, S. Shafer, A. Blake, and K. Sugihara, eds., pp. 16-23, IEEE Computer Society Press, Los Alamitos, CA, 1995.

[VOG 07] Wouter V. Vogel, Jorn A. van Dalen et al. “Evaluation of Image Registration in PET/CT of the Liver and Recommendations for Optimized Imaging”, J Nucl Med 2007; 48:910–919.

[WIN 09] Wintermark, M., Flanders, A. E., Velthuis, B., Meuli, R., Van Leeuwen, M., Goldsher, D., Pineda, C., et al. (2006). Perfusion-CT assessment of infarct core and penumbra: receiver operating characteristic curve analysis in 130 patients suspected of acute hemispheric stroke. *Stroke; a journal of cerebral circulation*, *37*(4), 979–85. doi:10.1161/01.STR.0000209238.61459.39

[WOJ 10] Julien Wojak, Elsa D. Angelini, and Isabelle Bloch. 2010. Joint variational segmentation of CT-PET data for tumoral lesions. In Proceedings of the 2010 IEEE international conference on Biomedical imaging: from nano to Macro (ISBI'10). IEEE Press, Piscataway, NJ, USA, 217-220.

[ZHA 09] Zhao B, James LP, Moskowitz C, Ginsberg M, Qin Y, Riely G, et al. “Evaluating Variability in Tumor Measurements from Same-day Repeat CT Scans of Patients with Non-small Cell Lung Cancer”, Radiology, 2009:252;263-272.

# 4 Task 3.3: Image processing functionality as a Service

## 4.1 Introduction

Advances in medical imaging technology have made it possible to routinely acquire high-resolution, three- or four dimensional images of the human anatomy using a variety of imaging modalities. Today the number of acquired 2D images per exam rates from 150 images for screening up to 700 to 3000 for the diagnosis of complex cases. This large amount of images per case together with the growing importance of medical imaging in clinical practice, have continuously increased the workload of the radiologist, which explains the need for computer –assisted medical image analysis. Furthermore there is a quest for objective, quantitative information from medical images. The use of 3D image processing and visualization techniques makes direct inspection of the scene in three dimensions feasible and greatly facilitates the extraction of quantitative information from the images. Appropriate *image processing techniques* can be applied to automate the measurement process so that *highly objective* and *reproducible results* are obtained. Improvements in image quality, changing clinical requirements, advance in computer hardware and algorithmic progress in medical image processing all have direct impact on the state of the art in medical image analysis.

Advanced image processing functionality is traditionally offered as part of a closed image processing workstation. The way in which it is to be proposed in MEDUSA requires a whole new approach where the functionality is presented as a service, which can be accessed externally. Image processing functionality as a service can be seen as a specialization of Software as a Service. Software as a Service (SaaS) is a software delivery model in which software and associated data are centrally hosted on the cloud [Was 11]. SaaS is typically accessed by users using a thin client via a web browser. The approach of SaaS starts to become common for clinical management applications, as explained in [AMA]. However, we could not find many references in the literature to existing image processing services. Most of the works describe prototypes that are not yet connected to clinical information systems, but which explore various solutions provided by cloud and grid computing to provide remote and distributed image processing or image access services.
For example, Chiang et al (2011) have produced a cloud-based platform using a Service-Oriented Architecture (SOA) to run scripts (Macro) that use the ImageJ image processing toolkit. This service enables running automated processing software (defined by Macro) as a service on cloud resources. A web-based prototype offers services to process X-rays of the hand for bone age assessment. Similarly, [Ojo 13] present a system for analysis of mammography with a web interface and which uses the Amazon EC2 cloud infrastructure.
[YAN 10] present a distributed medical image storage system based on the Hadoop Distributed File System, but the paper does not discuss its potential for image processing. [Gui 11] presents how a CT reconstruction algorithm can be implemented using web services that run on a distributed infrastructure on demand. To keep costs down, the hospital only maintains the resources needed for emergency CT scans. During burst periods, such as during the public opening hours of the CT laboratory, it relies on an infrastructure dynamically extensible with virtual machines from IaaS cloud infrastructures.[Nas 10] presents concepts for medical imaging services, and their discovery, based on a grid computing approach. It provides a good review of the

literature on this topic, but does not describe any concrete system. [Lan 13] revises the flow of images and data between the various actors and systems involved in the radiology process to reduce data transfers and be friendlier for remote and distributed computing, such as clouds. The Saas and SOA topic are discussed extensively in the appendix chapter 8.1.

The (thin) client–server model is a distributed application structure in computing that partitions tasks or workloads between the providers of a resource or service, called servers, and service requesters, called clients [Sun 09]. Often clients and servers communicate over a computer network on separate hardware, but both client and server may reside in the same system. A server is a host that is running one or more server programs which share their resources with clients. A client does not share any of its resources, but requests a server's content or service function. Clients therefore initiate communication sessions with servers which await incoming requests. Chapter 4.2 describes a cloud computing platform for the secure deployment and integration of the different services involved in a medical imaging process from bedside data collection to information distribution and remote access by medical staff.

The cloud computing platform mentioned in chapter 4.2 set some conditions which must be satisfied by processes running on it. To ease the migration of existing image processing applications a plug-in on the aforementioned platform will be adapted in the Medusa time frame and is presented in chapter 4.3.

When designed for a client-server environment, image processing and image browsing applications can place high demands on the underlying system and network resources. When interactivity is needed and the client to server connection spans a wide-area internet connection, the demands on the network can be extremely high. Chapter 4.4 addresses the problem of how to obtain interactivity by distributing the process load over the available resources.

Chapter 4.5 discusses the accessibility and full automation of image processing. Reduction of manual steps in protocols is necessary where instant actions are required, such as in acute care.

Finally an example of a cloud based service is discussed in chapter 4.6. This chapter contains an offering for the image storage, visualization and processing as a service in the domain of in-vitro diagnostic.

## 4.2 Application deployment and integration in the cloud

It is a widely accepted fact today that cloud computing provides an interface for the provisioning and exploitation of virtual resources, accessible on demand, and that can be controlled directly by the service owner. This allows consumers and businesses alike to be able to use complex applications without installation problems and with direct access their personal files from any computer on the internet. A description is given of a cloud computing platform for the secure deployment and integration of the different services involved in a medical imaging process from bedside data collection to information distribution and remote access by medical staff.

This platform called ACCORDS (**A**dvanced **C**apabilities for **C**ompatible**O**ne **R**esources **D**istribution **S**ervices) [MAR 12] , was developed under the CompatibleOne project [Com 09]. It offers a simple and unique interface allowing for the description of a

user's cloud computing needs, in terms of resources, and their subsequent provisioning on the most appropriate cloud providers. Resource descriptions may cover the complete cloud computing paradigm ranging from complete applications (SaaS), through development platforms (PaaS) down to low level compute, network and storage defined infrastructure (IaaS). It provides a unique language for the description and management of an unlimited number of cloud service providers, and its immediate benefits are:

- Centralized description of all cloud computing needs in terms of resources
- Rapid selection of the most appropriate provider
- Full automation of the provisioning and the migration process of applications and resources
- Elimination of any form of vendor lock-in and permit full interoperability
- Monitoring and control of the quality of service delivery of selected cloud providers

To achieve this, we propose the following high-level requirements for the systems engaged in the medical imaging process. Each application must be optimized in terms of computing resources:

- Automation of the installation process
- Description of required configuration
- Required Protocol
- Required security levels
- Hardware profile (CPU and GPU)
- Operating system

### 4.2.1 Design and lifecycle application in the cloud

The complete "lifecycle" of a cloud service includes both internal and external cloud resources, from request, through self-service provisioning, to decommissioning. Although some may equate the service lifecycle with the software development lifecycle, it comprises much more. This lifecycle is customized to the requirements of the business, with both the flexibility to deliver the required software stacks and the management framework to ensure the operational integrity of the cloud service.

Cloud lifecycle management ensures successful use of the cloud by implementing policy driven provisioning processes through a self-service portal for applications requirements description. With cloud lifecycle management in place, IT can achieve the fundamental goals of a cloud environment: agility, cost savings, and a more optimized use of resources (be they human beings, servers, or capital).
The key benefits of a cloud lifecycle management solution should include:

- Acceleration of the delivery of cloud services in response to business needs.
- Automation of provisioning and workflows, both for speed and cost effectiveness.
- Enabling users to request flexible configurable cloud services for their specific use cases.
- Supporting the use of public cloud infrastructures to augment and extend internal resources.
- Maximizing resource utilization by ensuring that all unused cloud services are recycled.

Initial decisions around cloud lifecycle management will help lay the foundation such that technological decisions progress, ensuring that the environment is flexible enough to address anticipated areas of growth in the future.

#### *4.2.1.1 ACCORDS platform cloud lifecycle*

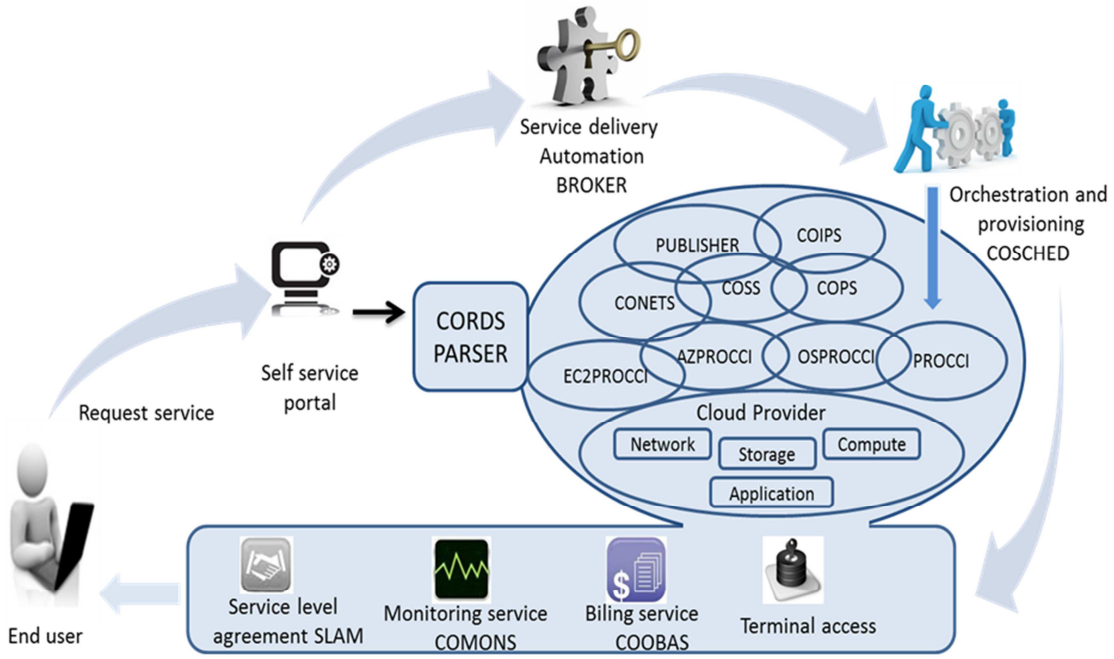


**Figure 19: ACCORDS platform cloud lifecycle**

Figure 19 shows the cloud lifecycle management of the ACCORDS platform. It supports the full lifecycle from the initial request through to service delivery and supervision.
The cloud lifecycle starts with a user description of the service that will be requested from the cloud. This description is based on the CORDS [Mar 08] descriptions schema. The Parser service will then be used to process this description. Once the parsing phase has been validated then a fully automated workflow will be invoked in order to perform placement of the requested service. Once this phase has completed then the service will be automatically provisioned.

According to the description the requested cloud service could require full-stack provisioning; server, storage, and network resources, as well as middleware, applications and other software elements, in order to be fully operational. This provisioning can be in any environment (virtual, cloud, or even physical) and may span server, network, and storage resources.

Once provisioned, the service enters its operational phase, where the normal day-to-day activities of performance, capacity, and compliance are managed with the dedicated services included in the ACCORDS platform such as service level agreement and monitoring.
When a cloud service is no longer required then the cloud resources constituting the service can be released.

All service components offered by the ACCORDS platform operate on-demand, at the user's explicit request, or according to a predetermined schedule.

## 4.3 Remote display solution on ACCORDS

### 4.3.1 Introduction

These days, all remote display solutions (wired or wireless, desktop computer or thin client oriented, Windows or Unix based, etc.) exploit the client-server architecture. Consequently, any remote display technological support can be assessed according to the following three criteria:

(1) Interception of visual content generated by the application at the server side.
(2) Compression and transmission of the content to the client.
(3) Management of the user interactivity (including the transmission of user events from client to server).

An additional fourth criterion related to the energy consumption is taken into consideration for mobile thin clients. The study in [CAR 10] brought to light that the energy consumption of a smartphone depends on the network (GSM/Wi-Fi), the CPU, the RAM, the display and the audio. While the last three factors are rather more related to the device and to the actual user behavior, the amount of data transmitted through the network and the CPU activity intrinsically depend on the technology and will be further investigated in our study.
In the next chapter different remote viewers versus operating system technologies are evaluated and rated on the previously mentioned criteria.

### 4.3.2 Remote desktop viewer / support technologies

This section considers the most frequently encountered remote desktop viewers and their support technologies (X window, NX, VNC, and RDP) for discussion according to the aforementioned four criteria.

#### *4.3.2.1 Unix / Linux X Windows*

The X window system represents the native remote viewer for all desktop Linux applications. The application output, intercepted by an XClient (running on the server), is represented by a rich set of 119 basic graphic primitives describing the X11 protocol. Such X content is transmitted to the XServer (running on the client) by the X11 protocol, which makes no provision for content compression. On the client side, the XServer not only displays the graphical content but also captures the user interactivity by generic Linux OS mechanisms (keyboard/mouse drivers). While ensuring good performance in wired desktop environments, the X window system cannot be directly employed for mobile thin clients, mainly because of its bandwidth and latency requirements; actually, no X window system for thin clients is currently available.

#### *4.3.2.2 NX technology*

By providing alternative protocols, the NX technology (a proprietary NoMachine solution) is intended to reduce the X11 network consumption and latency. Assuming an X window system is already running locally (the X Client, the X11 protocol and the X Server being all accommodated by the server), an NX proxy intercepts the X11 protocol, compresses it and subsequently transmits the result to the NX agent running on the remote client. Note that no particular user interactivity mechanisms have been

developed. Although the experiments showed a good compression rate of the initial X11 content, such a solution is not yet available.

#### *4.3.2.3 Microsoft Windows Remote Display Protocol (RDP)*

Microsoft Windows OS provides the RDP (Remote Display Protocol) framework, a proprietary client-server solution for remote displays, available in both desktop and mobile versions. On the server side, the RDP server intercepts the application output through the GDI (Graphical Device Interface) and represents it by a mixture of images, graphics and formatted text. This content is then transmitted using the RDP protocol to the RDP client where it is displayed. In desktop environments, the RemoteFX, an emerging extension of the basic RDP framework, also enables multimedia content transmission. The user interactivity is managed by RDP and/or Windows OS drivers. Although natively designed for the Windows OS, Linux-based RDP servers also emerged in the last months. Nowadays, the RDP clients target about 9% of the US smartphone market [BAN 11] and it is forecasted to have the most important relative growth by 2015 [GAR 12].

#### *4.3.2.4 Virtual Network Computing (VNC) remote viewer*

The VNC (Virtual Network Computing) remote viewer also assumes that an X window system is already available locally on the server side and brings new components in order to alleviate bandwidth and CPU constraints. The VNC Server (running on the server) intercepts the X graphical content at the XServer side. It further converts it to raw images (pixel maps) which are subsequently compressed and transmitted to the client by using the RFB (Remote FrameBuffer) protocol. On the VNC Client side (running on the client), the visual content composed only of images can be directly displayed. As in the previously discussed cases, the user interactivity is managed by the underling operating system. Several image compression optimizations are currently considered by VNC: TightVNC, TurboVNC, VNC-HEXTILE, VNC-ZRLE, … . For mobile thin clients, VNC-HEXTILE represents nowadays the most effective solution of that kind. Despite completely disregarding semantic information concerning the visual content to be displayed, VNC may be considered today as the most intensively used mobile thin client remote viewer: its myriads of versions for Linux, Andorid, iOS, RIM and even Windows Mobile can cover more than 87% of the personal smartphones in US [BAN 11].

#### *4.3.2.5 Semantic multimedia remote display for mobile thin clients*

The study in [BOJ 13] presented MASC, a multimedia oriented remote viewer. The principle consists of representing the graphical content as a real-time interactive multimedia scene-graph. The underlying architecture features novel components for scene-graph creation and management, as well as for user interactivity handling.

### 4.3.3 Remote viewer evaluation

The experimental setup considers the Linux X windows system and BiFS/LASeR multimedia scene technologies on the server and client sides, respectively. The implemented solution was benchmarked against currently deployed solutions (VNC and Microsoft-RDP), by considering text-editing and www-browsing applications.
The quantitative assessments demonstrate:

(1) Visual quality expressed by seven objective metrics, e.g. PSNR values between 30 and 42dB or SSIM values larger than 0.9999;
(2) Downlink bandwidth gain factors ranging from 2 to 60;

(3) Real-time user event management expressed by network roundtrip-time reduction by factors of 4 to 6 and by up-link bandwidth gain factors from 3 to 10;
(4) Feasible CPU activity, larger than in the RDP case but reduced by a factor of 1.5 with respect to the VNC-HEXTILE.

The performances exhibited by the remote display technologies presented above are synoptically illustrated in Figure 20.

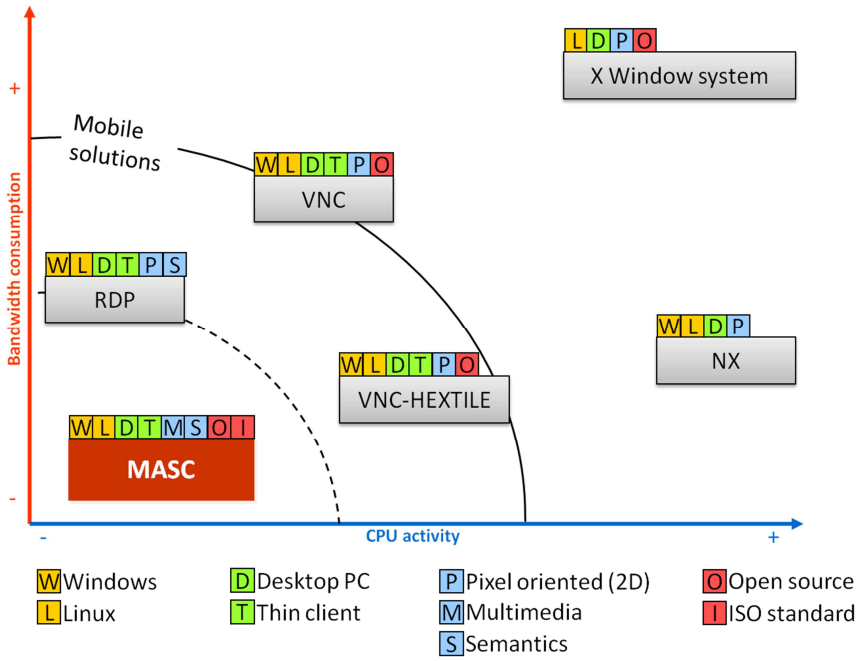


**Figure 20. Comparison of the current remote display solutions.**

### 4.3.4 Final note

In practice, the VNC and RDP technological supports are exploited by a large variety of ready to use applications. While an exhaustive list of such applications is practically impossible to be given and is also out of the scope of this document, consider for instance:

- VNC-based applications:
  - Apple Remote Desktop, Cendio ThinLinc, Chicken, ChunkVNC, Crossloop, EchoVNC, Ericom, Goverlan Remote Control, NoMachine NX, iTALC, KRDC, Mac HelpMate, N-central, noVNC, RealVNC, RapidSupport, Remote Desktop Manager, TigerVNC, TightVNC, TurboVNC, UltraVNC, X11vnc
- RDP-based applications:
  - AnywhereTS, Citrix XenApp, CoRD, DualDesk, Ericom, FreeRDP, NoMachine NX, KRDC, N-central, rdesktop, Remote Desktop Manager, Techinline, xrdp, XP/VS Server.

Moreover, proprietary (undisclosed) remote viewers' technological support and applications are also available.
For instance:

- TeamViewer exploits NAT (Network Address Translation) for establishing a connection, based on the RFB (VNC) and RDP protocols.
- GoToMyPC exploits the Citrix ICA (Independent Computing Architecture) proprietary protocol, but also supports VNC (RFB) and RDP.
- PhoneMyPC exploits its proprietary protocol, without exposing any specification detail.
- Oracle and Sun Microsystems offer Virtual Desktop Infrastructure (VDI) solution based on the proprietary ALP – protocol (Appliance Link Protocol).

Unfortunately, all these solutions are not open for research (undisclosed specification) and development (no source code); consequently, they cannot be objectively benchmarked in our research study.
All these applications may provide additional levels of functionalities, like built-in encryption, file transfer, audio support, multiple sessions, seamless window, NAT pass-through, IpV6 support, video or 3D. The study, reported in this chapter, is placed at the technological support level; consequently, the actual application peculiarities will not be taken into account.

## 4.4 Flexible image distribution over high-latency low-bandwidth channels

In this chapter we consider the effect of changing the typical client-server architecture to include monitoring of the bandwidth and latency parameters and to use these parameters to distribute the viewing pipeline processing steps
[Qis 08]. The viewing pipeline creates the 2D result image by applying the requested image processing steps. The viewing pipeline processing steps may be distributed over the server and the client. Thus dependent upon the client hardware configuration, the available bandwidth and the measured latency, part of the viewing pipeline may be implemented on the client.

Overall performance of a client-server environment is dependent on the bandwidth of the network between the client and server, latency of the application running on the server and the changes in the latency (jitter). Below a definition of bandwidth, latency and jitter are given.

- Bandwidth – The rate of data transfer measured in bits per second (bps)
- Latency – The average time per unit of data. Latency is the measure for the responsiveness for the user of the system and is measured in seconds.
- Jitter – The variation of latency.

To optimize performance it is necessary to make optimal use of the available resources. Also the client resources are taken into account. To effectively use the available resources it is necessary to continuously measure the parameters that vary and influence the performance, being the bandwidth, latency and jitter. The client and server hardware are known and their performance are assumed constant.

### 4.4.1 Bandwidth optimizations

The bandwidth of the connection between the client and the server is a varying and a limiting factor in the overall performance picture. As stated before, the bandwidth

determines the rate of data transfer per second. To achieve fast and efficient data transfer it is necessary to minimize the amount of data needed by the client application. Data reduction can be obtained with help of techniques like segmentation and compression. Higher compression rates can be reached for segmented images.

### 4.4.2 Image Segmentation

Segmentation is the process of partitioning a digital image into multiple segments (sets of pixels). The goal of segmentation is to simplify and/or change the representation of an image into something that is more meaningful and easier to analyze [Sha 01]. Image segmentation is typically used to locate objects and boundaries (lines, curves, etc.) in images. More precisely, image segmentation is the process of assigning a label to every pixel in an image such that pixels with the same label share certain visual characteristics. The result of image segmentation is a set of segments that collectively cover the entire image, or a set of contours extracted from the image (edge detection). Each of the pixels in a region is similar with respect to some characteristic or computed property, such as color, intensity, or texture. Adjacent regions are significantly different with respect to the same characteristic(s) [Sha 01]. Several segmentation methods have been proposed in medical images, e.g. algorithms based upon approaches such as histogram analysis, region growing, edge detection, and pixel classification. For more detail on image segmentation see chapter 3 Task 3.2: Use case specific image processing and modeling.

### 4.4.3 Image Compression

The objective of image compression is to reduce irrelevance and redundancy of the image data in order to be able to store or transmit data in an efficient form. Image compression may be lossy[5] or lossless[6]. Lossless compression is preferred for archival purposes and often for medical imaging. This is because lossy compression methods, especially when used at low bit rates, introduce compression artifacts. Effective loss-less compression algorithms for medical image have been developed which result in saving of storage space and better utilization of bandwidth and speed of data transmission [Kof 06], [Jan 11]. For more detail on image compression see chapter 3 Task 3.2: Use case specific image processing and modeling.

### 4.4.4 Latency optimizations

Latency, the responsiveness of a system is dependent upon the throughput of the system. The throughput of the system is defined as the average amount of data that can be processed per unit of time. There are multiple ways to achieve throughput targets. In many cases, it is possible to allocate more computing resources to the problem, in which different data units will be processed simultaneously by different parts of the computer system. This will often increase the throughput, but comes at the expense of the additional computer resources. The server will typically put an upper limit to the amount of additional computer resources that may be used.

In Image Processing, roughly two processing steps are performed on the server:

- Processing of the 3D or 4D medical volume data, resulting in e.g. a 2D image.
- Processing of the 2D result image to enhance it for optimal image interpretation.

[5] A data encoding method that compresses data by discarding (losing) some of it.
[6] A data encoding method that allows the exact original data to be reconstructed from the compressed data.

The client resource may also be considered as a computing resource. The first processing step where the potentially large volume data is involved cannot be offloaded to the client. The second processing step may be split into independent functional steps of which some can be performed on the client resource. To achieve this, the Processing pipeline that prepares the 2D result image for optimal image interpretation must be designed in such a way that nodes present in the processing pipeline can be distributed over different resources
[Qis 08] (see Figure 21).

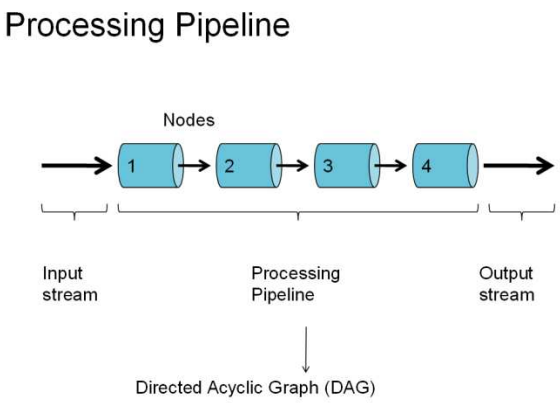


**Figure 21. Processing Pipeline**

The processing pipeline typically consists of nodes that perform different image processing steps on the input stream (e.g. blurring, edge enhancement, gray-level windowing, interpolation, rasterization[7]). Some or all processing steps can be performed on the client, thereby:

- Reducing the load on the server.
- Possibly avoiding a round trip to the server.

Caching of intermediate processing results may reduce computational overhead and increases performance at the cost of memory.
It must be noted that for processing of the result image on the client, the 2D result image as produced by the server may have a different format compared to the format of the 2D result image that would otherwise be produced. This difference may manifest itself in a larger 2D result image, which in turn has an impact on the network load (reduces the available bandwidth).

When the addition of computational resources is not feasible, other techniques must be explored to improve the throughput. These techniques include the optimization of algorithms such that these can be computed with fewer computational needs and the restructuring of algorithms such that these can be mapped more efficiently on computer hardware. The HiPiP ITEA 07022 project focused on getting the image processing result within the real-time requirements[Hi1 11] [Hi2 11], [Hi3 11], [Hi4 11] , [Hi5 11], [Hi6 11], [Hi7 11], [Hi8 11].

### 4.4.5 Jitter optimizations

There are a few reasons to consider jitter:

- In hard real-time systems the total of the latency plus the jitter determine the maximum response time of the system.

[7] Rasterization is the task of taking an image described in a vector graphics format (shapes) and converting it into a raster image (pixels or dots) for output on a video display or printer, or for storage in a bitmap file format.

- In user interaction, such as in eye-hand coordination during precision steering, jitter causes deterioration of control and therefore deterioration of accuracy.

In interaction, jitter is perceived as poor quality.
Thus in user interaction where precise steering is required, one scenario could be to measure the worst case latency and if this latency is still acceptable then artificially adapt the latency for all following interactions in order to keep it constant. A constant latency will allow humans to adapt to the pace of the application and is easily accepted. Variations in latency, i.e. jitter, is annoying.

### 4.4.6 Conclusion

Change the typical client-server architecture to include monitoring of the bandwidth and latency parameters and use these parameters to distribute the viewing pipeline processing steps
[Qis 08]. The viewing pipeline creates the 2D result image by applying the requested image processing steps. The viewing pipeline processing steps may be distributed over the server and the client. Thus dependent upon the client hardware configuration, the available bandwidth and the measured latency, part of the viewing pipeline may be implemented on the client.

## 4.5 Accessibility and full automation of image processing

The MEDUSA platform makes the latest technologies of medical image processing and cloud computing available for users. This opens new opportunities for the implementation of new business processes and services around sharing of images and accessing images.
One of the possibilities is the offering of available image processing methods as services that can be reached remotely. This is a valuable alternative for various medical centers, instead of buying complete software packages, including service contracts and workstations.

### 4.5.1 Full automation

A potential disadvantage of remote software functionality is the longer response time. This can be disturbing if a lot of user interaction is required for the image processing analyses. This is the main motivation to study the reduction of required user-interaction and to come to fully automated image processing. Another motivation is the need to reduce manual steps in protocols where instant actions are required, such as in acute care. The simplification and reduction of manual actions can reduce variability in the protocols.

In this chapter we will consider a couple of image processing applications and study how existing methods can be applied to remove required user-interaction for these applications such that they can be offered as a service. Below, we discuss possibilities to automate software execution to make it independent from user interaction. In many software analysis packages, the main manual intervention performed is the depiction of the region of interest, e.g. selection of arteries, selection of tumor, etc. We here describe automatic detection of the region of interest for further automation of the image processing method. This subject of image processing is also known as Recognition, which is the high-level process of determining roughly the whereabouts of an object of interest and distinguishing it from other object-like entities in the images.
Two types of activities in medical image recognition can be distinguished: Computer Aided Diagnosis (CAD) and Computer Aided Visualization and Analysis (CAVA). In

CAD the focus is on disease or lesion detection. CAVA deals with identification of anatomical structures. Recognition in CAVA does not have the purpose of determining the presence (or absence) of an object, but it is used for subsequent delineation and quantification. The most common method of recognition in current software practices for CAVA is the manual delineation by a user, for example by placing seed points or placing an initial model near the object. We here describe current methodologies for automated recognition for the purpose of removing manual interaction with the images.

Commonly reported recognition methods can be distinguished into 4 different approaches:

- Filter based approaches using filters such as the Hough transform [SAN 05], or marker-based watershed segmentation [LAP 02];
- Pattern recognition-based approaches using feature analysis after global segmentations (See Figure 22-right [ELA 13]);
- Heuristics-based. e.g. searching the center of gravity of a particular region of interest;
- Atlas-, (model-, or template-) based recognition. In this approach an image or a model of a structure is created and matched with patient data.

In the literature, the atlas based recognition is by far the most used approach. An atlas refers to a specific model for a population of images with parameters that are learned from a training data set [Cra 04]. An atlas can be built in an a-prior manner [Sca 95] or generated from training datasets [Coo 94]. Parametric atlas methods typically combine training images into a single atlas image, whereas non-parametric atlas methods use all training images separately. The simplest example of an atlas is a mean intensity image, which is also known as a template. An atlas can include richer information such as local image statistics and the probability that a particular spatial location has a certain label.

The simplest approach of using an atlas is to model a medical image as a deformed version of a single template. For example, MRI brain scans are often mapped to the MNI template [Eva 92]. A disadvantage of using templates is that large variations in patients cannot be mapped. Alternatively, multiple templates can be used. In this case an image is represented as a deformed version of a template, e.g. a template for healthy subjects and another for diseased patients. The optimal number of templates to use is still subject to research.

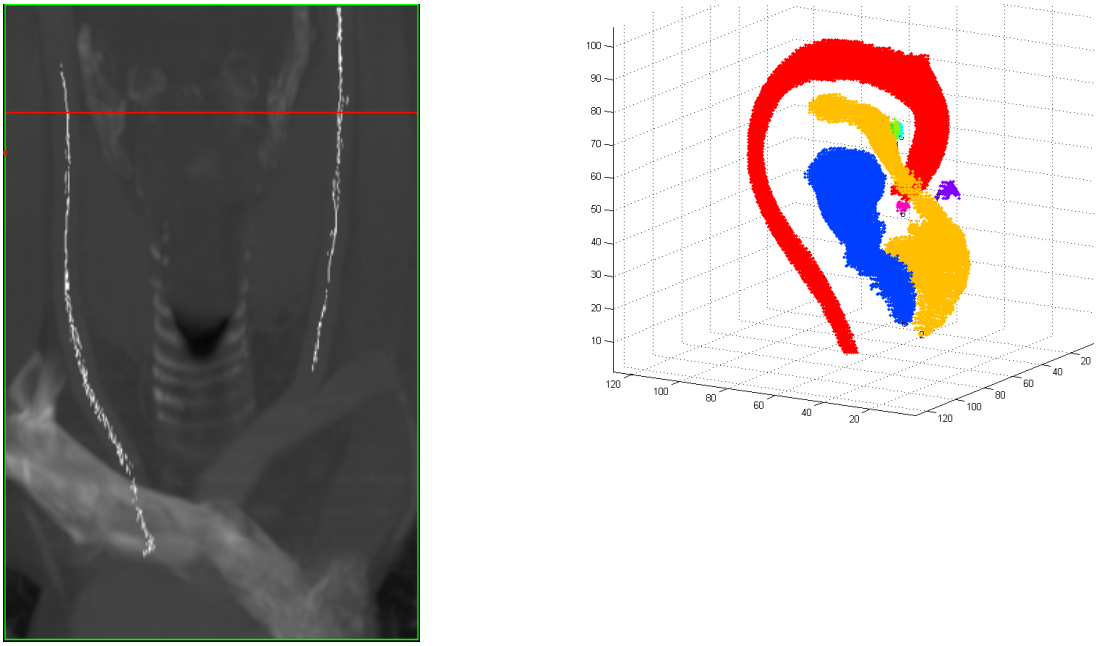

**Figure 22 - Left: Example of a filter based technique for the automated detection of the carotid arteries. In this example, a Hough transform is used for the detection of circle-like structures in every z-plane. Right: globally segmented objects. The recognition of the object of choice is performed using fuzzy voting of various features**

An important step in the process is the mapping between the search image and the template. After an initialization step, the corresponding positions can be determined using maximum correlation of the search image and template using intensity or other features.
Model based initialization can be performed with deformable models of which statistical shape models are the most popular.
An important aspect of the model based segmentations and initialization is the search of optimal parameters. Commonly this is performed by the combination of a global search (with e.g. evolutionary strategies) and a local search, which subsequently to optimize e.g. non-rigid shape parameters.

## 4.6 Image storage, visualization and processing as a service for in-vitro diagnostics

Currently, laboratories involved in the in-vitro diagnostic usually perform all the necessary steps (processing of the sample, staining, slide preparation, microscopic analysis and drawing the final conclusion) internally, with the visual microscopic analysis. We could identify only two kinds of service offer involving imaging. Of these, the first is clearly out of Medusa scope as it only deals with the data locally; it is mentioned here because one of the Imstar goals in Medusa project is offering similar services remotely, with distant results validation.

### 4.6.1 Complete service

The user sends the slides to be analysed to the service provider, who scans the slides using the in-house hardware (microscope or slide scanner), then performs the image analysis procedures and supplies the final report to the user. This type of offer is currently proposed by many companies, including Imstar. This kind of service requires the diagnostic expert (pathologist) to physically be at the same site as the slides to be

analysed. In some cases, to perform the requested analysis, the service provider has to develop relevant automated or semi-automated imaging procedures.

### 4.6.2 Virtual slide

The user sends the slides to the provider, who scans the slides, stores the scanned images at the server and provides a means for the user to visualize and annotate the images.
The key point here is the image size (from $10^7$ to $10^9$ of pixels) that requires the image visualization software able to load only the pixels required for the given visualization rectangle at the requested resolution. Multi-resolution image file format is required to support such functionality. Currently, there is no universally accepted standard for the file formats, neither for the high-resolution images nor for the annotations; every slide scanner manufacturer proposes its own file format with the corresponding SDK provided to OEMs. Attempt of Microsoft with its Deep Zoom technology does not seem to attract the slide scanner manufacturers (at least for the moment).
At the best of our knowledge there is currently no offer for automated image processing conducting to the diagnostic associated with the virtual slides.

## 4.7 References

[Ana 95] Patt Anal Mach Intell (1995) 17(6) 545-561
[AMA] American Medical Association, Is an application service provider (ASP) software or cloud computing service right for your practice?. URL: http://www.ama-assn.org/resources/doc/psa/software-as-service.pdf
[Arc 02] Archip, N. et al. “A knowledge-based approach to automatic detection of the spinal cord in CT images”, Medical Imaging, IEEE Transactions on (Volume: 21, Issue: 12), 2002.
[BAN 11] M. Bannan, D. Kellogg, “All about android”, Nielsen, 2011
[Blo 04] Bloomber, J. (2004). SOA + EDA = FUD. ZAPFLASH-06092004.
[Bla 04] Thomas Blaffert and Rafael Wiemker. "Comparison of different follow-up lung registration methods with and without segmentation", Proc. SPIE 5370, Medical Imaging 2004: Image Processing, 1701 (May 12, 2004); doi:10.1117/12.535345.
[BOJ 13] B. Joveski, M. Mitrea, P. Simoens, I.J. Marshall, F. Prêteux, B. Dhoedt, “Semantic multimedia remote display for mobile thin clients”, Multimedia Systems (MMSJ), vol. 19, no. 1 (February 2013).
[Bul 05] T. Bulow, R. Wiemker et al. “Automatic Extraction of the Pulmonary Artery Tree from Multi-Slice CT Data”, Proc. SPIE 2005 Medical Imaging Conference, SPIE vol. 5746, pp.730-740, 2005.
[Cha 05] Arnaud Charnoz, Vincent Agnus et al. “Liver registration for the follow-up of hepatic tumors”, MICCAI 2005, LNCS 3750, pp. 155-162, 2005.
[CAR 10] Aaron Carroll , Gernot Heiser, “An Analysis of Power Consumption in a Smartphone”, USENIX 2010, June 22–25, 2010
[Com 09] http://www.compatibleone.org
[Coo 94] Cootes T., et al.: The use of Active shape models for locating objects in medical images. Imag Vis Comp 12(16) (1994) 355–366.
[Cra 04] M. De Craene and A. B. d Aische and B. Macq and S. K. Warfield. "Multi-subject registration for unbiased statistical atlas construction". Proceedings of Medical Image Computing and Computer-Assisted Intervention 2004.
[Cry 88] Cryer HM, Miller FB, Evers BM, Rouben LR, Seligson DL. “Pelvic fracture classification: correlation with hemorrhage”. J Trauma 1988, 28:973-980.

[Dal 04] J.A. van Dalen, W. Vogel et al. “Accuracy of rigid CT-FDG-PET image registration of the liver”, Phys Med Biol. 2004; 49:5393–5405.
[ELA 13] Elattar, M.A, E.M. Wiegerinck, R.N. Planken, E. vanBavel, H.C. van Assen, J. Baan Jr, H.A. Marquering (2013) Automated Segmentation of Aortic Root in CT Angiography of Patients evaluated for TAVI Eligibility (Submitted for publication)
[Era 07] Eral, T. (July 2007). SOA, Principle s of Service Design.
[Eva 92] C. Evans and D. L. Collins and B. Milner, "An MRI-based stereotactic atlas from 250 young normal subjects", Journal Soc. Neurosci. Abstr. 18: 408, 1992.
[Gar 07] Garcia M, Jemal A, Ward EM, Center MM, Hao Y, Siegel RL, Thun MJ. “Global Cancer Facts & Figures 2007”, Atlanta, GA: American Cancer Society, 2007.
[GAR 12] “Forecasts for smartphone operating system sales until 2015”, Gartner, www.rossdawsonblog.com
[Gin 10] B. van Ginneken, S.G. Armato et al. "Comparing and combining algorithms for computer-aided detection of pulmonary nodules in computed tomography scans: the ANODE09 study", Medical Image Analysis 2010;14:707-722.
[Gui 11] Sam Guinea, Gabor Kecskemeti, Annapaola Marconi,and Branimir Wetzstein. Multi-layered Monitoring and Adaptation. In: Proceedings of ICSOC 2011. Springer Verlag, LNCS 7084, pp. 359–373, 2011.
[Hi1 11] WP4 partners, “Report on resource management for parallelisation”, ITEA 07022 HIPIP deliverable D1.3.3, pp. 7–10, 2011.
[Hi2 11] WP4 partners, “Report on resource management for parallelisation”, ITEA 07022 HIPIP deliverable D1.3.3, pp. 11–15, 2011.
[Hi3 11] WP4 partners, “Report on medical imaging parallelisation”, ITEA 07022 HIPIP deliverable D1.1.3, pp. 7–11, 2011.
[Hi4 11] WP4 partners, “Report on medical imaging parallelisation”, ITEA 07022 HIPIP deliverable D1.1.3, pp. 27–30, 2011.
[Hi5 11] WP1 partners, ITEA 07022 HIPIP deliverable D1.1.3 Consortium public, pp. 30, 2011.
[Hi6 11] WP2 partners, “Report on application parallelisation”, ITEA 07022 HIPIP deliverable D2.1.3, 2011.
[Hi7 11] WP2 partners, “Report on virtualization in application parallelisation”, ITEA 07022 HIPIP deliverable D2.2.3, 2011.
[Hi8 11] WP2 partners, “Report on Resource Management in application parallelisation”, ITEA 07022 HIPIP deliverable D2.3.3, 2011.
[Hil 04] Hildebrand, F., Giannoudis, P., Kretteck, C., et al. “Damage Control, Extremities”, Injury, Int. J. Care Injured (2004) 35, 678-689.
[Hub 09] Huber-Wagner et al. "Effect of whole-body CT during trauma resuscitation on survival: a retrospective, multicentre study", Lancet 2009. Apr 25;373(9673):1455-61.
[Int 09] Intel, I. (2009). Architecting Software as a Service for Entreprise.
[Jan 11] J. Janet, Divya Mohandass and S. Meenalosini (2011), “Lossless Compression Techniques for Medical Images In Telemedicine, Advances in Telemedicine: Technologies, Enabling Factors and Scenarios”, Prof. Georgi Graschew (Ed.), ISBN: 978-953-307-159-6, InTech,
[Kan 03] Y. Kang, K. Engelke, and W. A. Kalender, "A new accurate and precise 3-D segmentation method for skeletal structures in volumetric CT data", IEEE Trans. Med. Imag., vol. 22, no. 5, pp.586-598 2003.

[Kof 06] Koff DA, Shulman H. “An overview of digital compression of medical images: can we use lossy image compression in radiology?” Can. Assoc. Radiol. J. 2006 Oct; 57(4):211-7.

[Lan 13] Steve G. Langer & Ken Persons & Bradley J. Erickson & Dan Blezek. Towards a More Cloud-Friendly Medical Imaging Applications Architecture: A Modest Proposal. J Digit Imaging (2013) 26:58–64

[LAP 02] Lapeer, R, Tan A C and Aldridge, R. V. (2002) A combined approach to 3D medical image segmentation using marker-based watersheds and active contours: the active watershed method. Medical Image Understanding and Analysis

[Lee 87] Lee, E., & Parks, T. (1987). Synchronous data flow. Proceedings of the IEEE.

[Lee 95] Lee, E., & Parks, T. (1995). Dataflow process networks. Poceedings of the IEEE, vol 75.

[Lub 07] Meghan Lubner et al.” Blood in the Belly: CT Findings of Hemoperitoneum”, RadioGraphics 2007; 27:109–125.

[MAL 01] (Maldjian, Chalela, Kasner, Liebeskind, & Detre, 2001)

[MAR 12] Iain James Marshall, L01.3 - AccordsPlatform v1.2.pdf, 2012, CompatibleOne project.

[Mar 08] Iain James Marshall, Cords Reference Manual http://www.compatibleone.fr/occi/publisher/CordsReferenceManualV2.08.pdf

[Mur 11] Keelin Murphy, Bram van Ginneken et al. “Evaluation of Registration Methods on Thoracic CT: The EMPIRE10 Challenge”, IEEE TRANSACTIONS ON MEDICAL IMAGING, 2011.

[Nas 10] Aisha Naseer and Lampros K. Stergioulas. Web-Services-Based Resource Discovery Model and Service Deployment on HealthGrids. IEEE TRANSACTIONS ON INFORMATION TECHNOLOGY IN BIOMEDICINE, VOL. 14, NO. 3, MAY 2010

[Oas 06] OASIS, O. C. (2006). Reference Model for Service Oriented architecture 1.0.

[Ojo 13] Iuliana Ojog, Miguel Arias-Estrada, Jesus A. Gonzalez, Beatriz Flores. A Cloud Scalable Platform for DICOM Image Analysis as a Tool for Remote Medical Support. In: Proceedings of eTELEMED 2013 : The Fifth International Conference on eHealth, Telemedicine, and Social Medicine. p. 246-249

[Pap 02] Pape HC, Giannoudis P, Krettek C. “The timing of fracture treatment in polytrauma patients: relevance of damage control orthopedic surgery”. Am J Surg 2002;183:622—9.

[Qis 08] Qishi Wu, et al. “Self-Adaptive Configuration of Visualization Pipeline over Wide-Area Networks”, IEEE Transactions on Computers, Vol. 57, No. 1, January 2008.

[Ran 06] Rangaraj M et al.“Method for the automatic detection and segmentation of the spinal canal in computed tomographic images”, Proc. SPIE 2006 vol. 15, Journal of El. Imaging.

[REI 13] J.J. Reimerink, H.A. Marquering, A.C. Vahl et al. Semi-automatic sizing software in emergency endovascular repair for ruptured abdominal aortic aneurysms. Submitted, 2013

[Rik 09] E. M. van Rikxoort, B. J. de Hoop, M. A. Viergever, M. Prokop, and B. van Ginneken, “Automatic lung segmentation from thoracic computed tomography scans using a hybrid approach with error detection”, Medical

[SAN 05] Sanderse, M., Marquering, H. a, Hendriks, E. A., van der Lugt, A., & Reiber, J. H. (2005). Automatic initialization algorithm for carotid artery segmentation in CTA images. Med Imag Comput Assist Interv, 8(2), 846–53.

[Sca 95] Scarloff S. and Pentland A.: Modal matching for correspondence and recognition. IEEE Trans

[Sch 08] SCHMELZER, R. (Feb. 28, 2008). Why Service Consumers and Service Providers Should Never Directly Communicate. ZAPFLASH-2008228.

[SCH 11] M. Schrijver, P.C. van Haren et al. “Performance Focus Matrix”, ITEA 07022 HIPIP, 2011.

[Sha 01] Linda G. Shapiro and George C. Stockman (2001): “Computer Vision”, pp. 279-325, New Jersey, Prentice-Hall, ISBN 0-13-030796-3.

[Sha 11] Shanhui Sun, Christian Bauer et al. “Robust Active Shape Model Based Lung Segmentation in CT Scans”, LOLA11, 2011.

[SHA 12] Sharma & Venugopalan 2012

[SHI 12] (Shieh & Chang, 2012).

[Sil 11] José Silvestre Silva, et al. “Fast volumetric registration method for tumor follow-up in pulmonary CT exams”, Journal of Applied Clinical Medical Physics, Vol. 12, No. 2, Spring 2011.

[Sun 09] "Distributed Application Architecture". Sun Microsystem, Retrieved 2009-06-16.

[THE 00] Patrick Therasse et al. “New Guidelines to Evaluate the Response to Treatment in Solid Tumors”, Journal of the National Cancer Institute, Vol. 92, No. 3, February 2, 2000.

[Vio 95] P. Viola and W. M. Wells III, “Alignment by maximization of mutual information”, International Conference on Computer Vision, E. Grimson, S. Shafer, A. Blake, and K. Sugihara, eds., pp. 16-23, IEEE Computer Society Press, Los Alamitos, CA, 1995.

[Vog 07] Wouter V. Vogel, Jorn A. van Dalen et al. “Evaluation of Image Registration in PET/CT of the Liver and Recommendations for Optimized Imaging”, J Nucl Med 2007; 48:910–919.

[Was 11] "Software As A Service: Strategic Backgrounder". Washington, D.C.: Software & Information Industry Association. 28 February 2001. Retrieved 24 April 2011.

[Woo 07] M. Woodside, G. Franks, D.C. Petriu, "The Future of Software Performance Engineering", 2007.

[YAN 10] Yang et al, 2010] Chao-Tung Yang Lung-Teng Chen Wei-Li Chou Kuan-Chieh Wang. Implementation of a Medical Image File Accessing System on Cloud Computing. In: Proceedings of the 13th IEEE International Conference on Computational Science and Engineering. IEEE, 2010. P-321-326. DOI 10.1109/CSE.2010.48

[Zor 03] R. Zoroofi , Y. Sato et al. "Automated segmentation of acetabulum and femoral head from 3-D CT images", IEEE Trans. Inf. Technol. Biomed., vol. 7, no. 4, pp.329-343 2003.

[Zha 09] Zhao B, James LP, Moskowitz C, Ginsberg M, Qin Y, Riely G, et al. “Evaluating Variability in Tumor Measurements from Same-day Repeat CT Scans of Patients with Non-small Cell Lung Cancer”, Radiology, 2009:252;263-272. Physics, vol. 36, no. 7, pp. 2934–2947, 2009.

# 5 Task 3.4: Processing speed optimization

Medical image processing requires massive computation, in particular for images and signals. This computation makes medical image processing time consuming and providers of the medical imaging solutions almost always look for the ways to accelerate the image processing. Typical data volumes involved in the medical IP (hundreds of megabytes) hint for the parallelization as the principal speed optimization technique. Parallel processing is currently supported at multiple levels, ranging from multi-core CPUs via muliti-CPU machines to the computer clusters. Another relatively recent development is generalization of the use of specialized processors, notably that of graphical processing units (GPUs). The software industry offers the tools to simplify the parallelization at all these levels, which can be classified as follows:

1. Tools simplifying the parallelization at the computer level, such as Open MP. Alternatively, developers may use the means provided by the operating system API.
2. Tools provided by the manufacturers of the high-performance hardware, such as CUDA libraries/environment proposed by the GPU manufacturer NVIDIA. Currently, the developers rather use the OpenCL, a fruit of the standardization effort, that lets the application developer to target multiple hardware platforms.
3. Tools simplifying and standardizing the inter-computer communication and synchronization (message passing), such as OpenMPI, targeting the use of multiple computers (clusters, grids).

Some of the Medusa contributors previously worked for another EU funded project, HIPIP ITEA 07022), that was dedicated to the image processing parallelization. At the end of this project a method for analysis of the processing speed optimization needs has been proposed [SCH 11]. This method is based on separate consideration of the needs in throughput, latency and jitter. It is clear that for the applications involving real-time, such as acute care, the raw throughput of the optimization solution is insufficient, as the latency is critical.
In this chapter we present a review of the state-of-the-art of technologies and approaches that can be followed to achieve the image processing speed optimizations required by the medical image processing applications, namely OpenCL, OpenMP, distributed processing oriented for high throughput, and latency optimization, the two last cases represent work done within the HiPiP project.

## 5.1 OpenCL for medical image processing

OpenCL is a framework for writing code that run across heterogeneous platforms. This type of platform presents different processors such as central processing units (CPUs), graphics processing units (GPUs), digital signal processors (DSPs), etc. OpenCL is an open standard defined by the Khronos Group [KHR 13], the same organization that manages a wide variety of standards such as OpenGL, WebGL, and WebCL to name a few. The OpenCL standard defines a set of capabilities and features that the hardware vendors must meet to support parallel programming on their devices. The most notable vendors that have contributed and adopted the OpenCL standard are Intel, AMD, NVIDIA, Altera, and ARM. These vendors manufacture devices such as CPUs, discrete GPUs, FPGAs, and embedded chips.

OpenCL includes a language for writing kernels (functions that execute on OpenCL devices), plus application programming interfaces (APIs) that are used to define and then control the platforms. A benefit of OpenCL is that the kernels (parallel functions) that are coded according to this standard can run on different devices without any modification. This makes it possible to take advantage from computationally powerful devices that are well suited for different tasks. A GPU, for example, is a massively parallel device that can run SIMD instructions very efficiently, being optimized for floating point operations and matrix arithmetic. In terms of floating point operations, a GPU can achieve multiple teraflops of computational power, while a CPU has around hundreds of gigaflops of computational power. On the other hand, a CPU can handle more complex calculations with branching, recursion, and iterative algorithms. Consequently, this makes CPU and GPUs suitable for different purposes. To summarize, each device has suitable applications.

OpenCL is based on the C programing language with a few limitations and extensions. These limitations are recursion, function pointers, variable-length arrays, and bit fields. However, the kernel language is extended to provide vector types and operations, synchronization, and built in math functions to aid in concurrent programming. The OpenCL specification also defines an API to manage platforms, devices, memory, and execution. This API has a low level of abstraction and gives the developer the responsibility to correctly handle devices, memory, execution order, synchronization, etc. Figure 23, which is a class diagram of OpenCL objects, illustrates the complexity of handling multiple devices.

For example, a computer has two platforms, one is AMD with a GPU and the other is Intel with a fusion CPU with integrated GPU. Both support OpenCL, and provide computational power for different tasks. A developer who would like to use all available devices on this system would have to work with two different platforms. All the objects such as memory and events will not be interoperable with the other platform, since they are implemented differently in the underling driver. The developer will need to create support for this interoperability, together with execution synchronization between devices, as well memory consistency, etc.

OpenCL is a relatively new technology and there are no frameworks focusing on its application in the medical images processing context. Because of this, only high level frameworks that aim to make the development of OpenCL applications easier or more effective will be described further. Because they are designed to generic purpose development, they can also help in the medical image processing.

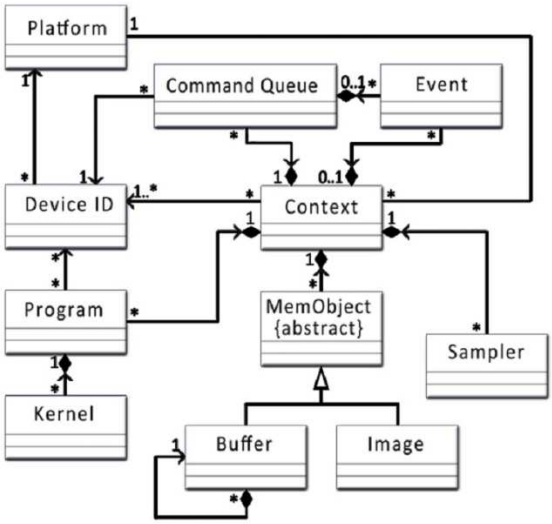


**Figure 23. The OpenCL UML class diagram. Platform: A platform is a vendor that supports OpenCL and its runtime environment. All other objects are implemented and handled by that vendor driver. Objects created by different platforms are not interoperable; Context: A context is an instance in one platform. All the other objects except for the device are instantiated from that context. There can be multiple contexts per platform, depending on the implementation strategy of the application developer; MemObject: A Buffer or an Image objects are allocated memory either shared in one context with all devices or specified for a single device; DeviceID: The DeviceID is an identifier of a device that is capable of running kernels; Program: A program is a compiled set of kernels that can be ran by a queue on a selected device; Command Queue: A command queue is an execution queue of kernels or memory transfers that are to operate on a given device. Multiple command queues on a given device can be used for concurrent operations, depending on the implementation strategy of the application developer; Event: Events are part of command queues that keep track of execution phases, used for synchronization and profiling.**

### 5.1.1 Kernel code generation

Some algorithms share similar algorithmic properties that can be expressed in so-called skeleton code, from which different algorithms can be easily generated. Skeleton code provides a template to fill in with specific operations to create a certain algorithm. Reduction algorithms for sum, minimum, or maximum, for example, are very similar, except for the final operation of adding to a total for sum, or comparing and swapping for the minimum or maximum. Research such as described by [ENM 10] and [NUG 11] explore these similar algorithmic properties, and provide simple function template constructs to generate and perform algorithms on accelerated devices

AMD has recently released a similar kernel code generating tool with function templates named BOLT [SAN 00]. BOLT has a number of built in functions, such as sort, reduce, and transform. These functions have an extra feature called *functors*, which are classes that implement an operator which transform data. With BOLT, these functors can be generated to CPU kernel code or OpenCL kernel code. BOLT has also extra features to determine if a dataset can better be processed on the GPU or CPU. A large drawback of BOLT is that it only runs in the Visual Studio environment for Microsoft Windows, and does not support Linux. It also does not provide extra task scheduling or work partitioning of large computations.

This approach provides effective kernel code generating features. However, more complex algorithms cannot be efficiently and elegantly implemented with skeleton code strategies. These solutions also focus more on kernel code generation than on effective execution strategies.

### 5.1.2 Expanding to clusteres

Many GPUs Package (MGP) [BA1 10] is a framework that enables to run OpenCL code on GPU equipped computer clusters. The framework was designed to cope with large number of computers and GPUs. MGP was built on top of the Virtual OpenCL layer (VCL) [BAR 00]. VCL is a transparent layer that accesses and manages accelerated devices on a cluster and presents these devices to a single node as if they were all on a single computer. VCL dispatches commands to the node of the remote device and handles the memory transfers between the nodes. MGL is a layer above VCL and adds extra rich functionality such as greater ease of programming by hiding low level API and incorporates built in tasks like scatter, exchange, and gather. MGL does not use extensive optimization strategies such as task scheduling or asynchronous execution and communication.

### 5.1.3 Task Scheduling

Research has already shown that is possible to speed up the execution of kernels on a GPU by saturating its computational units and memory bandwidth [GRE 10]. [SPA 10] have shown to increase performance by the usage of multi-buffering, by overlapping memory transfers, and also by overlapping the kernel execution. By splitting a large workload into smaller chunks, the data could simultaneously be transferred to the GPU, executed, and transferred back. This solution ties in the dataflow model and also in the task scheduling model. It follows a dataflow model because it streams independent execution and communication, and it follows a task scheduling model because the larger tasks are split into smaller subtasks and executed.

StarPU is a framework that takes advantage of this GPU saturating approach. StarPU is a runtime system capable of scheduling tasks over a heterogeneous platform. The virtual shared memory system keeps track of data copies on accelerated devices and features a data pre-fetching engine. Memory consistency is handled by the framework, which has a garbage collector mechanism to free unused memory. Combined with a database of self-tuned per-task performance models, StarPU can dynamically schedule computation tasks on multiple devices. It currently supports CUDA and OpenCL. The framework API uses the C programming language. In StarPU code readability and maintainability is reduced, and there is still a lot of overhead OpenCL code when using OpenCL functions.

.

## 5.2 Algorithm parallelization using OpenMP for medical image processing

OpenMP (Open Multi-Processing) is an API that supports multi-platform shared memory multiprocessing programming in C, C++, and Fortran on most processor architectures and operating systems, including Solaris, AIX, HP-UX, GNU/Linux, Mac OS X, and Windows platforms. It consists of a set of compiler directives, library routines, and environment variables that influence run-time behavior.

Historically, a key challenge in parallel computing has been the lack of a broadly supported, simple to implement parallel programming model. As a result, numerous vendors provided different models, with often mixed degrees of complexity and portability. Software programmers subsequently found it difficult to adapt applications to take advantage of multicore hardware advances. OpenMP was designed to bridge this gap, providing an industry standard, parallel programming API for shared memory multi-processors, including multicore processors.
OpenMP is managed by the nonprofit technology consortium OpenMP Architecture Review Board (or OpenMP ARB), jointly defined by a group of major computer hardware and software vendors, including AMD, IBM, Intel, Cray, HP, Fujitsu, Nvidia, NEC, Microsoft, Texas Instruments and Oracle Corporation.
OpenMP is an implementation of multithreading, a method of parallelizing whereby a master thread (a series of instructions executed consecutively) forks a specified number of slave threads and a task is divided among them. Then, these threads will run concurrently being allocated to different processors.
The section of code that is meant to run in parallel is marked with a preprocessor directive that will cause the threads to fork before the section is executed. After the execution of the parallelized code, the threads join back into the master thread, which continues onward to the end of the program.
The runtime environment allocates threads to processors depending on usage, machine load and other factors. The number of threads can be assigned by the runtime environment based on environment variables or in code using functions.

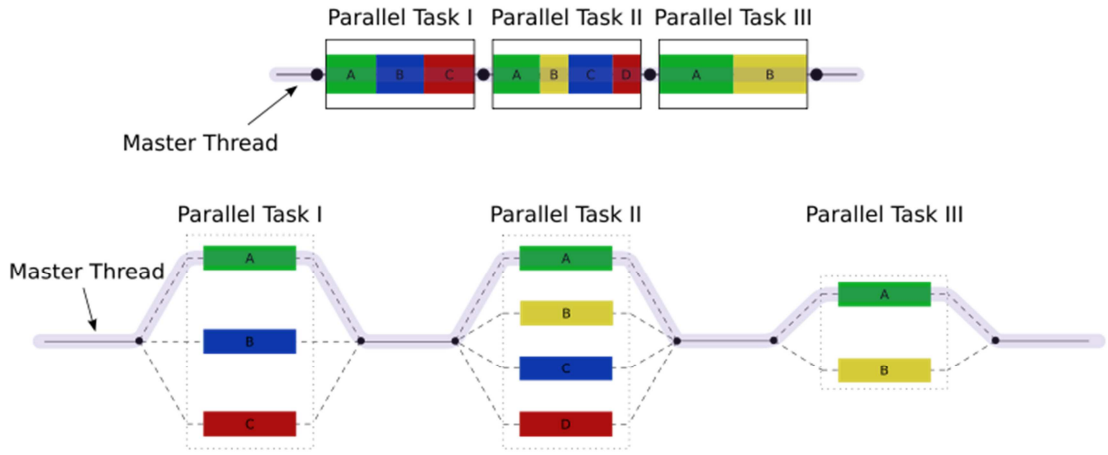


**Figure 24** ***Illustration of multithreading where the master thread forks off a number of threads which execute blocks of code in parallel. (http://en.wikipedia.org/wiki/OpenMP)***

The core elements of OpenMP are the constructs for thread creation, workload distribution (work sharing), data-environment management, thread synchronization, user-level runtime routines and environment variables.
In C/C++, OpenMP uses #pragmas. The OpenMP specific pragmas are detailed in Version 3.1 [OPE 11] of the OpenMP specification released July 9, 2011 and in [NOV 00]. OpenMP has been implemented in many commercial compilers. For instance, Visual C++ 2005, 2008 and 2010 support it, as well as Intel Parallel Studio for various processors. GCC has also supported OpenMP since version 4.2. GCC 4.7 supports OpenMP 3.1. One might expect to get an *N* times speedup when running a program parallelized using OpenMP on an *N* processor platform. However, this seldom occurs

for reasons detailed in [FUR 10]. Finally the main benefits and the drawbacks of OpenMP are summarized in Table 9 [http://en.wikipedia.org/wiki/OpenMP]. OpenMP is particularly suitable for medical image processing when large 2D and 3D matrix of data need to be passed through. [SLA 00] gives examples for spatial transformation and also for mathematical morphology, filtering and normalization. In each case using a quad-core CPU, the process time between the single-threaded code and the multi-threaded code corresponds to a speedup close to 3.5.

**Table 9: Benefits and drawbacks of OpenMP**

| Benefits | Drawbacks |
|---|---|
| Portable multithreading code in C/C++. | Risk of introducing difficult to debug synchronization bugs. |
| Simple: need not to deal with message passing as MPI does. | Currently only runs efficiently in shared-memory multiprocessor platforms. |
| Data layout and decomposition is handled automatically by directives and #pragmas. | Requires a compiler that supports OpenMP. |
| Incremental parallelism: can work on one part of the program at one time, no dramatic change to code is needed. | Cannot be used on GPU. |
| Unified code for both serial and parallel applications: OpenMP constructs are treated as comments when sequential compilers are used. | Reliable error handling is missing. |
| Original (serial) code statements need not, in general, be modified when parallelized with OpenMP. This reduces the chance of inadvertently introducing bugs. | Lacks fine-grained mechanisms to control thread-processor mapping. |

## 5.3 Image processing speed optimization for automated in-vitro diagnostics

### 5.3.1 Scope

Automated cell-based in-vitro diagnostic relies on image processing algorithms that detect the objects of interest (cells, nuclei, and, may be, other stained objects), then characterize and classify them. This task is performed without user intervention and has no real-time requirements. That is, we are concerned with the raw analysis throughput and not with latency and jitter minimization [SCH 11]. In some cases, means of visual review and validation of the detected and classified cells are given to the scorer.

The automated diagnostic process starts from the slide digitization phase: the slide is loaded into the microscope and the zone of interest (usually in the order of 1cm²) is scanned using the motorized stage, focus motor and camera controlled by the application. The digitized slide represents an image set totaling between 100 and 5000 MP (millions of pixels). The scanning speed is in the order of 2MP / sec.

The principal speed optimization requirement for such a system arises from the fact that processing speed is significantly (3-20 times) slower than that of the digitization. Thus, to ensure the diagnostic instrument can run continuously we have to accelerate the processing speed until it matches that of the slide digitization.

### 5.3.2 Parallelization

Considering that processing of slides is mutually independent (i.e. processing of a slide does not in any way influence processing of any other one), the straightforward solution is to parallelize the processing of image sets corresponding to the individual slides. For the parallelization solution to be scalable beyond the number of CPU cores available in one computer, the solution should be distributed.
Further, increasing number of image sets processed in parallel at certain point would start to saturate the image I/O bandwidth. In this case, the image storage should be distributed between multiple computers, and the processing computers should at first pick the processing tasks corresponding to the images stored locally (i.e. on the processing machine).
The three-component architecture presented below satisfies the above requirements and provides the nearly-linear scaling to tens of the CPU cores. The components communicate via TCP/IP. The corresponding system has been specified, designed and implemented within the HIPIP project.

- Workstation: application module applying the indicated image processing protocol to the indicated slide. This application stays idle until it receives the processing request. The request has two parameters: the name of the images set to be processed and the name of the protocol to apply. The images to be processed are loaded and processed one by one. The parallelization is achieved by the running multiple instances on the workstation.
- Server: application module receiving processing requests and distributing them between available workstations. When formulating the processing request, the user does not assign the request to a particular instance of the workstation; the request is sent to the (unique) server that automatically decides which of the running workstations suits best and directs the processing request to it. When taking the decision, the server takes the affinity between the workstation processor and image storage into account.
- GUI (Control/Monitor): application module that allows sending processing requests, visualization of the processing queue, and some user control over the protocol execution, such as remove/suspend/resume the task.

### 5.3.3 Use of optimized low-level libraries

Applications that perform image processing by CPU spend significant time in the low-level code. Appropriate choice of the compiler and low-level libraries can increase the application performance significantly. Thus, switching from the MS Visual Studio 2010 C++ compiler to the one from Intel XE 2011 yields 2 to 3 times increase in the processing speed (for Imstar PathFinder application). Further, our preliminary experience shows that using the routines from Intel Performance Primitives library, such as the ones for FFT and convolution, is also a major factor in the performance optimization.

## 5.4 Latency improvements for real-time advanced image processing

### 5.4.1 Introduction

This paragraph contains a summary of the results obtained within the HiPiP project.
The Full Project Proposal of the HiPiP ITEA 07022 describes its scope as:

The management of protocols, scheduling and resources to optimize utilization (bandwidth, CPU) for a given multi-core hardware set with respect to advanced image processing.
Special attention was given to:

- Additional low-latency requirements for minimal invasive intervention.
- Gain of the parallelization of individual algorithms.
- Tools that measure, analyze, and predict gains.
- Platform software design for optimized parallel processing methods.

This summary is organized as follows. In section 5.4.2 the results are presented obtained during the investigation for parallelization of algorithms. Section 5.4.3 describes the conclusions reached in the resource management phase of the project. Finally section 5.4.4 briefly touches upon Software Performance Engineering (SPE) [Woo 07]. The concept and an example of SPE is described in a paper called 'Performance Focus Matrix, *A tool to ease communication on performance engineering*' [SCH 11] that was written as a byproduct of HiPiP.

### 5.4.2 Parallelization of Algorithms

#### *5.4.2.1 2D Image Processing Algorithms in Interventional X-Ray*

We determined that the parallelization mechanisms needed for multi-PC environments under low-latency constraints is quite different from that for intra-PC environments. So, an effective implementation (framework) distinguishes between the two and offers different mechanisms for each.

- The Windows process/thread scheduling can be effectively bypassed by pinning worker threads to selected processor cores, and by assigning tasks to these worker threads by a specially written scheduling thread.
- Under low-latency constraints, data partitioning is highly preferable over task/algorithm partitioning. As long as each algorithm in a pipeline of algorithms can be data-partitioned to a sufficient level (compared to the number of processing cores available), executing the algorithms in the pipeline consecutively gives the lowest latency. Put differently, using all processing power (cores) to process a single image gives the best latency performance, provided all cores can be kept busy.
- One drawback of a pure data partitioning approach: it increases the susceptibility to jitter, as all cores have to wait for the other cores to finish before being allowed to start executing the next algorithm or image. In the presence of disturbing processing (e.g. other applications or the operating system itself), this may lead to considerable idle time.
- The most effective granularity of data partitioning depends on the type of algorithm:
    - Point operations are best scattered over a large number of tasks to decrease jitter, but with each task containing sufficient image pixels to utilize the cache effectively.
    - The data partitioning of multi-resolution operations is determined by the amount of overlap needed. Due to this (typically large) overlap, and the relative complexity of such algorithms (reflected in the total [Hi5 11]compared to the whole pipeline), the number of data parts should be a small multiple of the number of (homogeneous) processing cores available. E.g. 16 or 32 data parts to be processed by 8 cores.
    - Segmentation algorithms are very difficult to parallelize, and are therefore best separated from the low-latency pipeline (i.e.

asynchronous operation). Such separation can be accomplished by assigning one or more cores dedicated to the segmentation task, and absorbing the segmentation result in the low-latency pipeline whenever it comes available (decoupling). Such a decoupling technique is very similar to “parallel applications”, as addressed by [Hi6 11], [Hi7 11], and [Hi8 11].

### 5.4.3 Resource management

#### *5.4.3.1 Results of resource management for multi-core processor platform*

In a performance design we have focused on bandwidth reduction and latency optimization. Because of the memory-rich processing algorithms, straightforward data partitioning is unable to exploit a full locality of data. This has led to substantial inefficiencies in performance and waste of memory bandwidth. Therefore, a combination of data partitioning and functional partitioning was proposed. As a result, a substantial performance improvement was realized by a memory-communication model that incorporated the memory and compute requirements of the video processing in two ways:

- An inter-function partitioning is chosen to satisfy the required compute resources across the processor cores and optimized for communication between functions.
- An intra-function partitioning is chosen for each function separately, so that caching will be exploited for low latency.

The simulated values for memory communication were validated on a multi-core processor platform processing 2K×2K images in real time at 15 Hz. Results showed a considerable memory-bandwidth reduction of 70% and a latency reduction of 40-70%, compared to straightforward data-parallel implementations. The difference between the performance prediction and measured performance was within 10-20% in all cases. The proposed techniques (e.g. cache-aware partitioning) have proven to be valuable for regular signal processing functions, such as filtering and noise reduction. However, if processing becomes more irregular, such as in analysis, the proposed partitioning algorithm has to become more adaptive and flexible to preserve the efficiency gain [Hi1 11] .

#### *5.4.3.2 Results of resource management for multi-GPU template matching*

**ARGOS** (Automated Recognition of Geometries, Objects, and Segmentations) is a 3D template fitting capability that helps to localize macromolecules in their native cellular context and relate orientation properties of these molecules to their environment. It combines high resolution molecular structure information that was determined by single particle analysis with 3D cellular context from tomography. Various test of ARGOS running on BULL hardware (multi-GPU architecture) were conducted to determine the scalability of the algorithm towards more GPU’s and thus identify potential bottlenecks in the implementation.

Although ARGOS is capable of running hybrid multiple GPU configurations, we concluded that we did not fully understand all the runtime details of multi-GPU configurations to predict the benefit of a certain configuration. This has led us to the conclusion that it is most appropriate to deliver the final end-customer product with a single GPU, since there the cost/benefit ratio is most unambiguously determined. In the future, if needed, multi-GPU configurations can be delivered, but careful testing of the performance gain for each combination needs to be performed in a way comparable to what was described in this chapter [Hi2 11]

### 5.4.4 Software Performance Engineering

The authors in [SCH 11] present the Performance Focus Matrix, a tool to ease communication on performance engineering of software applications. It facilitates in reaching understanding and agreement on the scope and abstraction of performance engineering. The matrix helps in achieving that goal by clearly identifying why certain techniques are being considered or applied and by identifying what is the stage of the project in reaching its target including possible bottlenecks. The matrix implements the Plan-Do-Check-Action cycle for continuous process improvement horizontally (see Figure 27) and discriminates vertically between four focus levels: Functionality, Single Application, Multi Application and Performance in changing environments (see Figure 26).

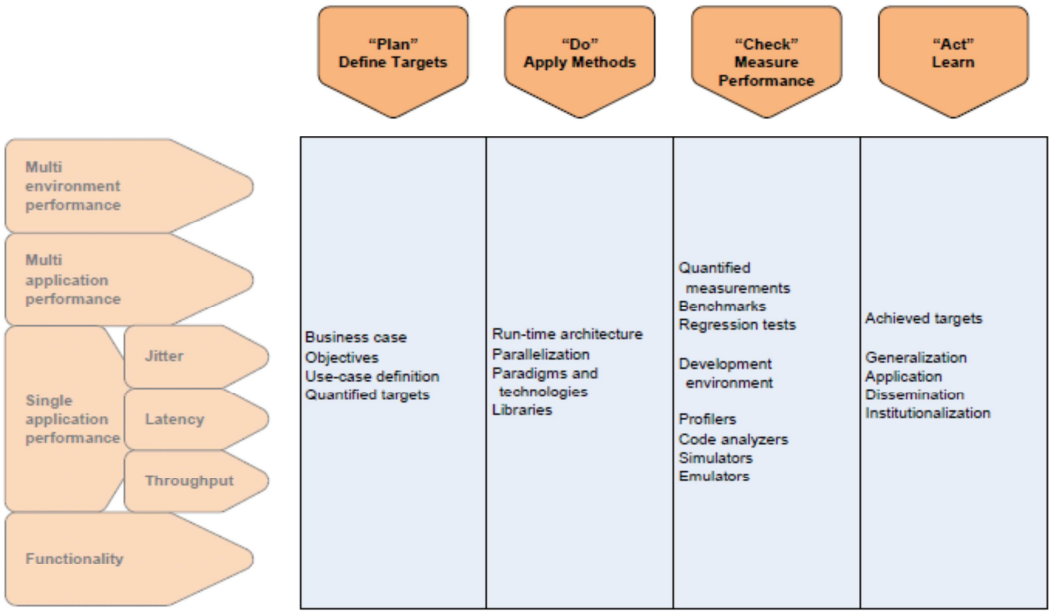


**Figure 25. The horizontal axis of the Performance Focus Matrix**

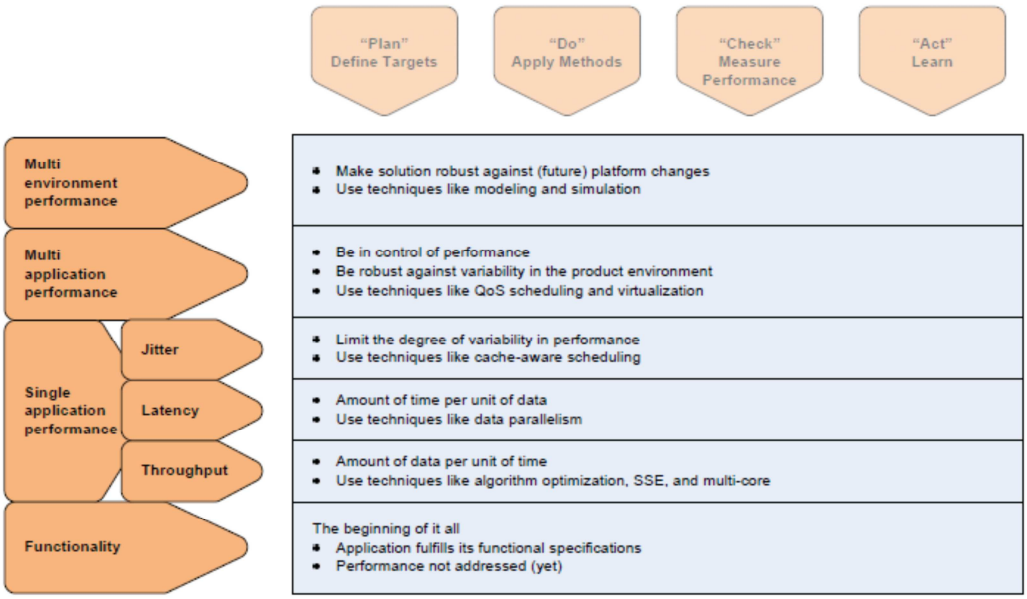


**Figure 26. The vertical axis of the Performance Focus Matrix**

## 5.5 References

[BA1 10] Barak, and Shiloh. 2010. "The Virtual OpenCL (VCL) Cluster Platform." http://www.mosix.org/txt_vcl.html/.

[BAR 00] Barak, A, and A Shiloh. n.d. "The Virtual OpenCL (VCL) Cluster Platform." http://www. mosix.org/txt_vcl.html.

[ENM 10] Enmyren, J, and C.W. Kessler. 2010. SkePU: A Multi-Backend Skeleton Programming Library for Multi-GPU Systems. Baltimore.

[FUR 10] Fürlinger K. OpenMP Application Profiling – State of the Art and Directions for the Future. ScienceDirect, Procedia Computer Science 00 (2010) 1-ger. 2010.

[GRE 10] Gregg, Dorn, Hazelwood, and Skadron. 2012. Fine-Grained Resource Sharing for Concurrent GPGPU Kernels

[KHR 13] Khronos Group website: http://www.khronos.org/

[OPE 11] OpenMP Architecture Review Board. OpenMP Application Program Interface. Version 3.1 July 2011.OpenMP. 2011. "OpenMP."

[NOV 00] OpenMP and automatic parallelization in GCC. Red Hat Canada.Novillo

[NUG 11] Nugteren, C, and H Corporaal. 2012. Introducing Bones: A Parallelizing Source-to-Source Compiler Based on Algorithmic Skeletons. Eindhoven.

[SAN 00] Sander, B. n.d. "GPU and CPU Programming Webinar." https://vts.inxpo.com. Slabaugh. n.d.

[SCH 11] M. Schrijver, P.C. van Haren et al. "*Performance Focus Matrix*", ITEA 07022 HIPIP, 2011.

[SLA 00] Slabaugh G., Boyes R., and Yang X. Multicore Image Processing with OpenMP. Medicsight PLC

[SPA 10] Spafford, K, J Meredith, and J Vetter. 2010. Maestro: Data Orchestration and Tuning for OpenCL Devices.

# 6 Task 3.5: Information Integration

## 6.1 Data integration Background

Demand for the integration and sharing of large amounts of data is growing due to the rapid increase in available sources of data coming from a large number of heterogeneous and distributed data sources, and a huge diversity of users.

The term "*data integration*" has a wide range of uses and interpretations. Gartner [TED10] defines the discipline of data integration as "practices, architectural techniques and tools for achieving consistent access to, and delivery of, data across the spectrum of data subject areas and data structure types in the enterprise to meet the data consumption requirements of all applications and business processes". Medical image processing is concerned with the data integration because managing large medical image collections is an increasingly demanding and important issue in many hospitals and other medical settings. A huge amount of this information is generated daily requiring robust and agile systems.

Unfortunately, today data integration systems are still extremely difficult to build and very costly to maintain. To deal with this problem, data integration research has moved from addressing architectural issues [GAR97, LEV96, UII97] to optimization of query execution [IVE99, KNO98, LAM99, FRI97, AVN00], and now onto optimization of system development, and other integration and sharing scenarios (e.g., data warehouses, peer data sharing, Web service composition), all toward the goal of making data integration systems widely practical. Many problems still are underlying this research as well as meeting the autonomic computing challenge. Hence, this research has the potential to make substantial advances in these areas. Most recently Cloud computing, which is used frequently in descriptions of the Web, is touted as a data integration pathway and a solution to improve the previous systems of information integration.

Cloud computing represents a true democratization of Web computing. It is not only changing the business models and the way in which IT infrastructure is being delivered and consumed, but also the underlying architecture of how we develop, deploy, run and deliver applications.

## 6.2 Cloud computing for data integration

### 6.2.1 Cloud Technology

Cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction [MEL10, MEL09].

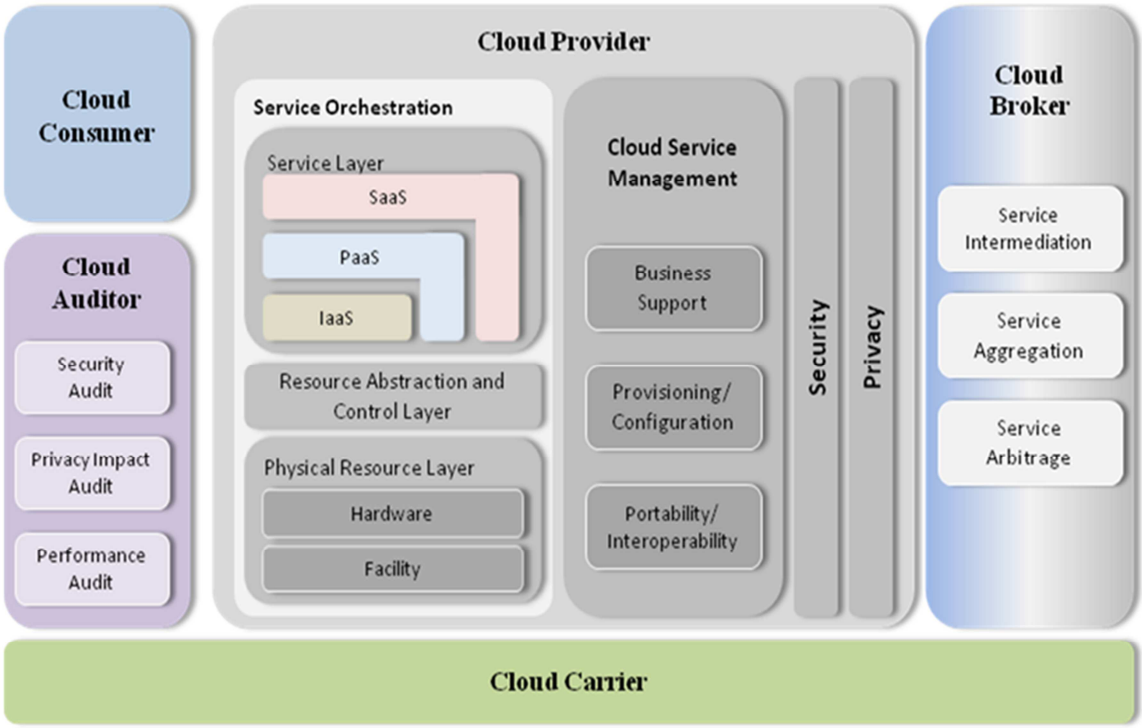


**Figure 27: Overview Taxonomy of Cloud Computing**

Given the promise of cloud computing, it is worthwhile to understand what is really happening inside a cloud (Figure 27). A cloud includes applications, infrastructure and a software environment or platform. All clouds do not include all of these levels, some provide simply the infrastructure. This figure gives the reader an understanding of what is inside the amorphous “cloud”.

#### *6.2.1.1 Architecture Components*

**Cloud application** (Software as a Service – SaaS). This layer provides capability for consumers to use the provider’s applications running on a cloud infrastructure. For instance, the applications are accessible from various client devices through a thin client interface such as Web browser. The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, storage, or even individual application capabilities. In this type of cloud service model, the security and privacy protection is provided as an integral part of the SaaS.

**Cloud software environment** (Platform as a service – PaaS). This layer offers capability for consumers to deploy consumer- created or acquired applications written using programming languages and tools supported by the cloud provider. The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, or storage, but has control over the deployed applications and possibly application hosting environment configurations. In this type of cloud service model, two levels of protection for security and privacy are required. At the lower system level, the cloud provider may provide basic security mechanisms such as end-to-end encryption, authentication, and authorization. At the higher application level, the consumers need to define application dependent access control policies, authenticity requirements, and so forth.

**Cloud infrastructure** (Infrastructure as a Service – IaaS). This type of cloud service model provides the capability for consumers to provision processing, storage, networks, and other fundamental computing resources, in which consumer is able to deploy and run arbitrary software, including operating systems and applications. The consumer does not manage or control the underlying cloud infrastructure, but has control over operating systems, storage, deployed applications, and possibly limited control of select networking components (e.g., host firewalls).

#### *6.2.1.2 Cloud Computing Actors*

As shown in Figure 27, the NIST [NIS12] cloud computing reference architecture defines five major actors:

**Cloud consumer** is the principal stakeholder that uses the cloud computing services. A cloud consumer represents a person or organization that maintains a business relationship with, and uses the service from, a cloud provider. A cloud consumer browses the service catalog from a cloud provider, requests the appropriate service, sets up service contracts with the cloud provider, and uses the service. The cloud consumer may be billed for the service provisioned, and needs to arrange payments accordingly.

**Cloud provider** is the entity (a person or an organization) responsible for making a service available to interested parties. A cloud provider acquires and manages the computing infrastructure required for providing the services, runs the cloud software that provides the services, and makes the arrangements to deliver the cloud services to cloud consumers through network access.

**Cloud auditor** is a party that can perform an independent examination of cloud service controls with the intent to express an opinion thereon. Audits are performed to verify conformance to standards through a review of objective evidence. A cloud auditor can evaluate the services provided by a cloud provider such as security controls, privacy impact, and performance.

**Cloud broker:** as cloud computing evolves, the integration of cloud services can be too complex for cloud consumers to manage. A cloud consumer may request cloud services from a cloud broker, instead of contacting a cloud provider directly. A cloud broker is an entity that manages the use, performance, and delivery of cloud services and negotiates relationships between cloud providers and cloud consumers.

**A cloud carrier** acts as an intermediary that provides connectivity and transport of cloud services between cloud consumers and cloud providers. Cloud carriers provide access to consumers through network, telecommunication, and other access devices.

#### *6.2.1.3 Deployment models*

**Public cloud** applications, storage, and other resources are made available to the general public by a service provider. These services are free or offered on a pay-per-use model. Generally, public cloud service providers like Amazon AWS, Microsoft and Google own and operate the infrastructure and offer access only via Internet (direct connectivity is not offered).

**Private cloud** is cloud infrastructure operated solely for a single organization, whether managed internally or by a third-party and hosted internally or externally. Undertaking a private cloud project requires a significant level and degree of engagement to

virtualize the business environment, and requires the organization to reevaluate decisions about existing resources. When performed right, it can improve business, but each and every step in the project raises security issues that must be addressed to prevent serious vulnerabilities.

**Hybrid cloud** is a composition of two or more clouds (private, community or public) that remain unique entities but are bound together, offering the benefits of multiple deployment models. Such composition expands deployment options for cloud services, allowing IT organizations to use public cloud computing resources to meet temporary needs. This capability enables hybrid clouds to employ cloud bursting for scaling across clouds. Cloud bursting is an application deployment model in which an application runs in a private cloud or data center and "bursts" to a public cloud when the demand for computing capacity increases and reversely back to the local cloud when the demand decreases.

### 6.2.2 Cloud data integration

Data integration can notably benefit from cloud computing because accessing multiple data sources and integration of instance data are usually expensive tasks. The Cloud computing paradigm enables organizations to take advantage of popular cloud computing models, such as SaaS, PaaS, and IaaS, while retaining a significant level of control over their solutions. Data integration systems, deployed on a cloud, offer compelling advantages in cost, quality of service, and agility by providing self-service access, elastic scalability, and usage metering. It also allows organizations to have complete control over the levels of performance and availability of service that they provide, and can easily enforce data governance regulations and auditing policies.

With the development of cloud computing, utility managers have now a third option, that of pushing some of the costs and risks associated with the management of data centers out to a third party. In December 2011, CSC [CSC11] released a report following a survey that it had conducted on 3600 companies in over 8 countries following the switch to cloud computing. The following list shows the key findings highlighted in this study.

- The switch to cloud computing was driven by the need to facilitate information access from multiple computing devices.
- The majority of businesses saved money. The savings varied by country and industry.

The concept of cloud computing fills a perpetual need of IT concerning data integration: a way to increase capacity or add capabilities on the fly without investing in new infrastructure, training new personnel, or licensing new software. Cloud computing encompasses any “subscription based” or “pay-per-use” service that extends IT’s existing capabilities in real time over the Internet.

## 6.3 Medical imaging in the cloud

Medical image processing applications require management of huge amounts of data and executive high performance computing. The management and analysis of this large amount of data effectively presents an enormous challenge for healthcare organizations as they struggle to manage access and share this data whilst trying to ensure cost effective operations.

In this context healthcare organizations are beginning to consider cloud computing as an attractive option for helping them manage medical imaging data.

### 6.3.1 Healthcare IT and cloud computing

In the healthcare sector, several existing projects include a research track dedicated to cloud computing. These pathways mainly investigate the migration possibilities from existing solutions to cloud based ones.

Among these projects, the **VISION Cloud** project identifies that cloud computing offers cost flexibility. In the past, a company or user with high storage needs would have to buy servers or even build a data center. Today, there are already many options allowing applications to be simply moved to cloud platforms where payment is only required for that which is really used. According to the description available in the Software and Services FP7 Project Portfolio, the objective of the **MIDAS** project is to realize a comprehensive framework able to support automation and intelligent management of Service Oriented Architecture (SOA) testing. MIDAS framework is made available as a Platform as a Service (PaaS) on a cloud infrastructure so as to support the elastic scalability of the testing environment, i.e. allocation of huge amounts of computation resources for relatively short test campaigns on very large services architectures. **CELAR** project recognizes that auto-scaling resources are one of the top obstacles and opportunities for research in Cloud Computing. This project directly addresses the gap of scalability that is important to healthcare applications as well. **Stratosphere** is a German research project (DFG-funded) investigating "Information Management on the Cloud" and creating the Stratosphere System for Big Data Analytics. The **e-Health GATEway to the Clouds** Project established a cloud-based research platform on the White Rose Grid to support e-health records research. There is also an ongoing call in Finland for development of a nationwide patient database through a cloud solution. This would include both data structure standardization in a national level as well as interoperability of hospital software and infrastructure.

Those research and development initiatives show that around the globe, healthcare reform has mandated that it is time for healthcare information technology (HIT) to be modernized and cloud computing is at the center of this transformation. The healthcare industry is shifting toward an information-centric care delivery model [CSC12], enabled in part by open standards that support cooperation, collaborative workflows and information sharing. Cloud computing provides an infrastructure that allows hospitals, medical practices, insurance companies, and research facilities to tap improved computing resources at lower initial capital outlays. Additionally, cloud environments will lower the barriers for innovation and modernization of HIT systems and applications. Cloud computing caters to the key technology requirements of the healthcare industry:

- Enables on-demand access to computing and large storage facilities that are not provided in traditional IT environments.
- Supports big data sets for electronic health records (EHR), radiology images and genomic data offloading, a burdensome task, from hospital IT departments.
- Facilitates the sharing of EHRs among authorized physicians and hospitals in various geographic areas, providing more timely access to life-saving information and reducing the need for duplicate testing.

- Improves the ability to analyze and track information (with the proper information governance) so that data on treatments, costs, performance, and effectiveness studies can be analyzed and acted upon.

### 6.3.2 Developed Outcomes and Applied Solutions

In addition to the work described in the above research initiative projects, there are also several other emerging solutions. These solutions illustrate the migration of existing software (applications, services, and data) from local computers or local servers to the cloud computing services.

**eMix**, which stands for Electronic Medical Information Exchange, is a cloud computing-based technology for secure sharing of medical imaging studies and reports between disparate healthcare facilities and physicians. eMix was developed to address a challenge in medical imaging: How to exchange medical imaging data between proprietary information technology (IT) systems that do not “talk to each other.”, The service also provides an alternative to legacy solutions typically used for sharing medical imaging data. eMix was one of the first cloud-based systems for accomplishing these tasks.

**AT&T Medical Imaging and Information Management** is a vendor-neutral, cloud-based service that combines AT&T Synaptic Storage as a Service$^{SM}$ with Acuo Technologies® Universal Clinical Platform™. The service provides access, viewing, storage and sharing of medical images from disparate picture archive communications systems (PACS) sold on an as needed, pay-as-you-go model. With this centralized solution, you can scale your image archive infrastructure in a highly secure, high availability operating environment - on an as-needed basis.

**Dell Medical Archiving (PACS, RIS/HIS)** it is a Clinical Data Management system specialized in medical image archiving. This Clinical Archival includes on-premise and cloud deployment options that can be used independently or as a hybrid, ensuring disaster recovery and instant scalability.

**SeeMyRadiology** is a cloud-based medical imaging platform for the sharing of medical images. It is the first solution that provided hospital integrated delivery network with the versatile, and proven technology to support its full range of needs, from image communications for trauma transfers and tertiary referrals to sharing files with referring physicians and patients.

**Medscribler** is a SaaS solution for recording patient data. It uses mobile technologies such as iPad and smartphones and handwriting recognition software to allow easy submission of patient data. It is an EMR solution that provides a quick and intuitive way to update medical records of patients. These records can also be stored in a cloud.

**LifeImage** is also a commercial solution for cloud-based image exchange. LifeIMAGE allows physicians, hospitals and patients to securely exchange images, anywhere. It provides simple workflow for managing exams from discs, receiving exams electronically from any outside source, integrating outside images with local systems, and sharing exams with outside physicians and patients.

**EMC2** Provides two services designed for healthcare sector: the “Collaborative Healthcare Solutions” and “Electronic Health Record Infrastructure”. EMC

Collaborative Healthcare Solutions include the EMC Medical Image Sharing and Management Solution, which provides a patient-centric infrastructure to "content-enable" Picture Archiving and Communication System, Hospital Information System, and Electronic Medical Record applications for accessing all relevant clinical, financial, and operational data. Based on open standards, the solution is in accordance with the Integrating the Healthcare Enterprise initiative that promotes the coordinated use of established standards.

**Canadian Infoway Health Ltd** identifies several key points in adopting cloud computing in the healthcare sector. They state that there are many opportunities for the application of cloud computing and have a broad range of possibilities. Such include virtualization of infrastructures for individual organizations into a private cloud; establishing public clouds for health promotion or decision support; or development of community clouds to support integrated care delivery models for chronically ill patients or to rapidly implement IT solutions for health care and health promotion programs.

The solutions presented in this section are only a part of the existing offers and illustrate the aspects in which cloud computing can be beneficial to medical imaging and to a healthcare professional in general.

### 6.3.3 Considering the Cloud for Medical Imaging

It is widely recognized that cloud computing and open standards are important cornerstones to streamlining healthcare. Cloud computing offers significant benefits to the healthcare sector [CSC11]: doctor's clinics, hospitals, and health clinics require quick access to computing and large storage facilities which are not provided in the traditional settings. Moreover, healthcare data needs to be shared across various settings and geographies, which further burden the healthcare provider and the patient, causing significant delay in treatment and loss of time. Cloud caters to all these requirements, thus providing the healthcare organizations an incredible opportunity to improve services to their customers, the patients, to share information more easily than ever was possible before, and improve operational efficiency at the same time.

- Electronic Medical Records. Hospitals and physicians are starting to see cloud-based medical records and medical image archiving services coming on line. The objective is to offload a burdensome task from hospital IT departments and allow them to focus on supporting other imperatives such as EMR (electronic medical records) adoption and improved clinical support systems.
- Telemedicine. With the increase in availability of mobile technologies and intelligent medical devices, telemedicine has grown to include not only tele-consultations and tele-surgeries, but also health record exchange, video-conferencing, and home monitoring. Cloud computing and the related ease of services deployment and data storage is an enabler for telemedicine.
- Big Data. Healthcare organizations turn to cloud computing to save on the costs of storing hardware locally. The cloud holds big data sets for EHRs (electronic health records), radiology images and genomic data for clinical drug trials. Attempting to share EHRs among facilities in various geographic areas without the benefits of cloud storage could delay treatment of patients.
- Health Information Exchange**.** Health information exchanges (HIE) help healthcare organizations to share data contained in largely proprietary EHR systems. Organizations may accelerate the deployment of HIE via a linkage to a strategic cloud implementation.

For the healthcare industry, cloud services represent an enormous opportunity. Storing, archiving, sharing and accessing images in the cloud allows the industry to manage data more efficiently and cost-effectively while overcoming many of the legal, regulatory, security and technical challenges that data requirements pose. The cloud enables hospitals to:

- Efficiently handle large bandwidth images;
- Use non-proprietary, standards-based, vendor-neutral architecture;
- Expand or contract storage capacity easily as needed;
- Manage authentication, encryption and security protocols;
- Conduct efficient system-wide application upgrades;
- Extend the life of existing infrastructure/investments;

### 6.3.4 Conclusion

Cloud services offer healthcare an attractive solution, helping hospitals scale with ease, better manage resources, and provide fluid access and sharing of medical images across organizations, departments and providers achieving a level of "*connectedness*" that supports healthcare organizations' patient care goals. Medusa project is an opportunity to demonstrate to Hospitals that they should carefully weigh the benefits and implications of cloud medical imaging and the special considerations that can make ascending into the cloud a true success.

## 6.4 Overview of the current state of meta-information integration into the in-vitro diagnostic process.

### 6.4.1 Introduction

Automated in-vitro diagnostic process, as implemented in the PathFinder platform and described in the T3.2-relevant chapter of this document, deals, in addition to images, with other data necessary to perform and validate the diagnostic. Under the meta-information integration we understand the ways to set, store, modify and report these data in the in-vitro diagnostic system. Below we overview the current organization of the meta-information and indicate its limitations.

### 6.4.2 Meta-information classification

The information involved in the in-vitro process, can be separated into the following categories:

#### *6.4.2.1 Reference base:*

List of the users and their rights (user rights are set to enable/disable display, annotation, validation, drawing final conclusion etc).

- List of slide preparation techniques;
- List of sample processing procedures;
- List of the valid conclusions (diagnostics) to be associated with a sample;

#### *6.4.2.2 Image set information*

- Sample identification (patient ID, date of sampling, nature of the sample, technician);
- Slide identification;
- Image capture conditions;

- Image analysis parameters.

#### *6.4.2.3 Detected cells and cell classification information (for every cell)*

- Cell contour;
- Nucleus(i) (and, may be, other labeled objects) contours;
- Cell parameters relative to the diagnostic (area, shape, nucleo-cytoplasmic ratio, optical density of the specific labeling etc.);
- Cell class, as assigned by the automated analysis. Set of classes depends on the diagnostic domain, in the simplest case it can be reduced to just two, say, “normal” and “alert”;
- Cell class, as assigned by each reviewer (the same image set can be analyzed by multiple pathologists).

#### *6.4.2.4 Diagnostic information*

- Resume of automated analysis;
- Conclusion set (the system allows conclusions to be made by multiple pathologists);
- Slide annotations.

### 6.4.3 Meta-information entry and storage

PathFiner system stores information either in the DB based on SQL server, or in files associated with the images. The access to the information stored in files is locked at the slide level. The Reference base information is set fixed at the moment of the system configuration; for the end user it appears as read-only. It is stored in the DB. Sample (incluing Patient when necessary) and Slide information are entered prior to the imaging and stored in the DB. Image capture conditions are stored in the image files using extended TIFF tags. Image analysis (protocol) parameters are stored in the text files; the files are referenced in the DB. Object contours are generated automatically during the image analysis protocol application; they are stored in dedicated binary files, associated with every image. Technically, the PathFinder applications allow changing the contours interactively, but in practice this feature is seldom used, the goal of the automated diagnostic system being automation.
Cell parameters and classification information are generated automatically, in the course of the analysis protocol, and stored, like the contours, in the dedicated binary files associated with the image files. Cell classification established by a reviewer is stored like the automated cell classification. Resume of the automated analysis is set during the analysis protocol application, conclusions are set by the user; both are stored in the DB. Slide annotations represent combination of outline (arrows, geometric shapes or freeform drawings) and text; they are set by the user and stored in the binary files associated with the image set.

### 6.4.4 Limitations

The principal limitation of the current scheme is availability of the global search, selection and filtering only for the part of the information stored in the DB. That means necessity to load all the information stored in files to perform a search/filtering operation on it. For example, that for an expert review of all the alert cells from a study of, say, 1000 slides, the application has to explicitly load all the classification information for all the slides from the study to filter it and present to the reviewer only the alert cells.

Another limitation is the absence of fine-grain locking of the classification information, which limits the concurrent access to this data.

## 6.5 References

[1] Ted Friedman, Mark Beyer, and Eric Thoo, —Magic Quadrant for Data Integration Tools,‖ *Gartner*, 19 Nov 2010.
[3] A. Y. Levy, A. Rajaraman, and J. Ordille. Querying heterogeneous information sources using sourcedescriptions. In Proc. of VLDB, 1996.
[4] Jeffrey D. Ullman. Information integration using logical views. In Proceedings of the 6th International Conference on Database Theory, Delphi, Greece, January 1997. Springer, Berlin
[5] Z. Ives, D. Florescu, M. Friedman, A. Levy, and D. Weld. An adaptive query execution system for data integration. In Proc. of SIGMOD, 1999.
[6] C. Knoblock, S. Minton, J. Ambite, N. Ashish, P. Modi, I. Muslea, A. Philpot, and S. Tejada. Modelingweb sources for information integration. In Proc. of the National Conference on Arti¯cial Intelligence (AAAI), 1998.
[7] E. Lambrecht, S. Kambhampati, and S. Gnanaprakasam. Optimizing recursive information gatheringplans. In Proc. of the Int. Joint Conf. on AI (IJCAI), 1999.
[8] M. Friedman and D. Weld. E±ciently executing information-gathering plans. In Proc. of the Int. Joint Conf. of AI (IJCAI), 1997.
[9] R. Avnur and J. Hellerstein. Continuous query optimization. In SIGMOD '00, 2000.[10] Peter Mell and Tim Grace, —The NIST Definition of Cloud Computing, Version 15, 10-7-09, 1, http://csrc.nist.gov/groups/SNS/cloud-computing/, accessed 1 Dec 2010.
[11]Cloud definitions from Mell and Grace, —The NIST Definition of Cloud Computing,‖ 2 and Cloud Computing Use Case Discussion Group, —Cloud Computing Use Cases,‖ white paper, 31 July 2009, 6, http://groups.google.com/group/cloud-computing-use-cases.
[12] CSC (2011) CSC Cloud Usage Index Report // http://www.csc.com/newsroom/ds/75354-ahead_in_the_cloud_the_csc_cloud_usage_index
[13] Cloud Standards Customer Council , “Impact of Cloud Computing on Healthcare”, November 2012.
[14] http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf
[AVN00] R. Avnur and J. Hellerstein. Continuous query optimization. In SIGMOD '00, 2000.
[CSC11] CSC (2011) CSC Cloud Usage Index Report,
[CSC12] Cloud Standards Customer Council , “Impact of Cloud Computing on Healthcare”, November 2012.
[FRI97] M. Friedman and D. Weld. Eficiently executing information-gathering plans. In Proc. of the Int. Joint Conf. of AI (IJCAI), 1997.
[GAR97] H. Garcia-Molina, Y. Papakonstantinou, D. Quass, A. Rajaraman, Y. Sagiv, J. Ullman, and J. Widom. The TSIMMIS project: Integration of information Heterogeneous sources. Journal of Intelligent Inf. Systems, 8(2), 1997.
[IVE99] Z. Ives, D. Florescu, M. Friedman, A. Levy, and D. Weld. An adaptive query execution system for data integration. In Proc. of SIGMOD, 1999.
[KNO98] C. Knoblock, S. Minton, J. Ambite, N. Ashish, P. Modi, I. Muslea, A. Philpot, and S. Tejada. Modeling web sources for information integration. In Proc. Of the National Conference on Arti¯cial Intelligence (AAAI), 1998.

[LAM99] E. Lambrecht, S. Kambhampati, and S. Gnanaprakasam. Optimizing recursive information gathering plans. In Proc. of the Int. Joint Conf. on AI (IJCAI), 1999.
[LEV96] A. Y. Levy, A. Rajaraman, and J. Ordille. Querying heterogeneous. information sources using source descriptions. In Proc. of VLDB, 1996.
[MEL09] Cloud definitions from Mell and Grace, —The NIST Definition of Cloud Computing,‖ 2 and Cloud Computing Use Case Discussion Group, Cloud Computing Use Cases,‖ white paper, 31 July 2009, 6, http://groups.google.com/group/cloud-computing-use-cases.
[MEL10] Peter Mell and Tim Grace, The NIST Definition of Cloud Computing, Version 15, 10-7-09, 1, http://csrc.nist.gov/groups/SNS/cloud-computing/ Dec 2010.
[NIS11] NIST, The NIST definition of cloud Computing, September 2011.
[TED10] Ted Friedman, Mark Beyer, and Eric Thoo, —Magic Quadrant for Data Integration Tools,‖ *Gartner*, 19 Nov 2010.
[UII97] Jeffrey D. Ullman. Information integration using logical views. In Proceedings of the 6th International Conference on Database Theory, Delphi, Greece, January 1997. Springer, Berlin.

# 7 Conclusions

The extent of this document indicates that medical image processing is an active field in computer science. We have addressed only a few topics that were the most relevant for MEDUSA.
Two topics were based on the actual functionality of image processing: (1) semantic based image compression and (2) use-case specific image processing, and three topics were based on the architecture and implementation: (1) image processing as a service, (2) high-performance image processing, and (3) information integration.

Chapter 2 deals with image compression. There are many approaches described for the compression of image data. Some are very generic whilst others use specific image constraints like the supposed symmetry of the human anatomy. For the Medusa implementation an important constraint is that the time required compressing and decompressing the data plus the time to transfer the compressed image should be less than the time required to transfer the original data. Experiments are needed to test whether this requirement can be met.

Chapter 3 Deals with use-case specific image processing functionality. The increased availability of medical image data and high-performance computers enables new functionality in medical practice. For example, for stroke imaging functionality for assessing the thrombus and early ischemic changes are currently not yet available. These methods have the potential to improve and ease the fast diagnosis and treatment of these acute patients. This chapter furthermore describes current status of hemorrhage detection, image processing for trauma related injuries, various methods for tumor segmentation. For multi-mode image analysis registration is important. The current state of the various current registration techniques is also described in this chapter. This chapter ends with the description of the current state of art in in-vitro imaging and image analysis.

The change of image processing software architecture towards software as a service is described in chapter 4. Advanced image processing functionality is traditionally offered as part of a closed image processing workstation. Software as a Service (SaaS) is a software delivery model in which software and associated data are centrally hosted on the cloud. Various approaches to deliver image processing as a service are detailed in this chapter, where the high latency over low bandwidths is the main problem to deal with. Recent developments have shown that cloud computing is currently the optimal approach for integrating heterogeneous data. However, issues remain to end up with a fully autonomous solution.

Chapter 5 addresses high performance image processing. Clinical advances have resulted in an increase of patient image data. These huge amounts of data can now currently be processed using new computer hardware. Parallelization on any scale (from multiple CPU's to multiple machines in a grid) is key in the advent of high-demand medical image processing. The introduction of tools to use these parallel (and GPU) resources enables the implementation of medical image analysis algorithms.

The current state of the art of information integration is presented in chapter 6.The demand of integration and sharing of multiple sources of data is rapidly growing. Data integration is defined as "practices, architectural techniques and tools for achieving consistent access to, and delivery of, data across the spectrum of data subject areas

and data structure types in the enterprise to meet the data consumption requirements of all applications and business processes". Huge amount of medical information is generated daily requiring robust and agile systems. Current research concerning data integration is mainly aimed onto optimization of system development, and other integration and sharing scenarios. Chapter 6 describes current state of art of using cloud architectures to deal with these issues.

# 8 Appendix

## 8.1 Designing SaaS and SOA applications

This section describes guidelines, principles and technologies for Service-Oriented Architecture (SOA) and Software as a Service (SaaS) applications [Int 09] .

### 8.1.1 Evolution of applications design patterns

When the use of the Web for business applications began to catch on in late 1995, a distributed application architecture (then sometimes referred to as an N-tiered architecture) emerged. This looked different from the traditional client-server architecture where the database generally sat on a centralized server and the thick client containing the user interface and application logic was installed on the desktop of each user. There were various flaws in this model, including scalability issues due to the dedicated database connections from each user as well as the operational overhead of the distribution, installation and maintenance of the client software on users' desktops.

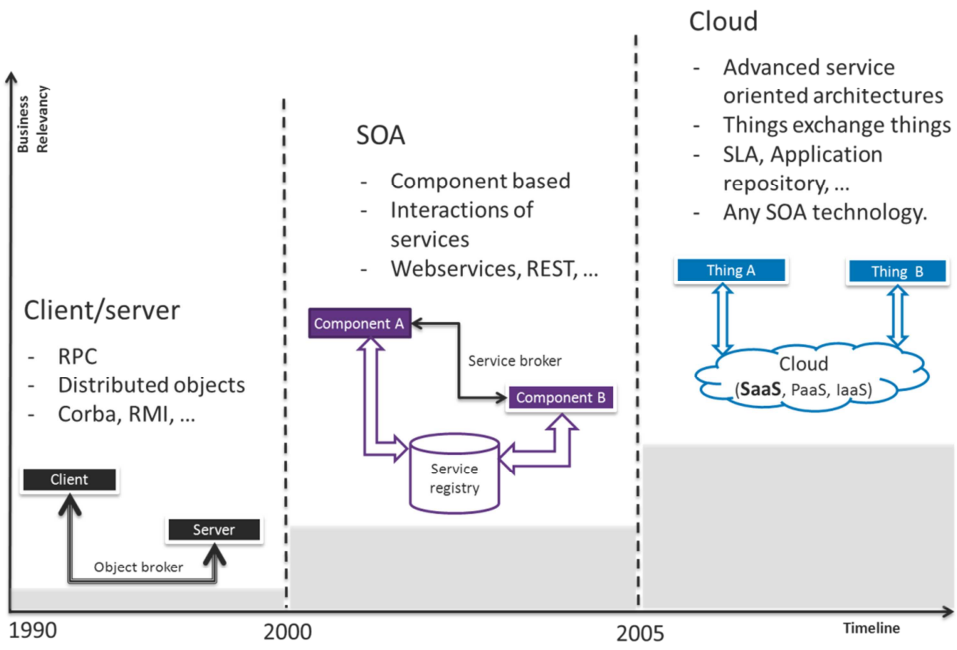


**Figure 28: evolution enabling SaaS**

The distributed application enhanced by Service Oriented Architecture was really made possible by two things. First, the Web browser emerged as a ubiquitous tool on the desktop. Second, a class of software known as an application server emerged that allowed rich application functionality to be delivered to the desktop via the Web browser used as a broker between users components(front office) and the application server components(back office). Gone were the distribution and maintenance costs as the client (Web browser) was now either installed within the operating system, or at least freely available for simple download and install. Also gone were the scalability issues because the application server environment split up the various components handling user requests so they could be scaled as needed.

**But then what is the difference between SOA and SaaS?**
It's widely accepted that Service Oriented Architecture is a means of designing and building software. It is a manufacturing model.

While Software as a Service is a mean of receiving software through an external party to your business similar to telephone or power utilities. Therefore it is a sales and distribution model.
So, both SOA and SaaS are hot topics in IT as both deliver in very specific ways material reduction in costs of IT operations and deliver a more agile alignment between business needs and IT solutions. The manner in which those benefits are delivered is the key difference. And in many ways, to capture the majority of the cost reduction and agility benefits it is often best to incorporate both SOA and SaaS in the IT landscape.

**Why incorporating SOA and SaaS?**

The main reason is that it makes sense to in-source as well as out-source some aspects of the modern business including IT. As mentioned above, SOA provides a means to deploy and quickly re-configure as business conditions change the applications, databases and other infrastructure hosted in data centers owned by a company. Recent versions of many departmental and enterprise application stacks from major software providers deliver products with a SOA design and implementation. Often ERP functions such as order management, general ledger, and manufacturing planning are still predominately delivered in this fashion.

While SaaS provides a readily accessible means to out-source the applications and databases (as well as some cases the functions) of a business. Some key examples include sales force automation, payroll, expense reporting and recruiting. One of the key considerations with SaaS is how to effectively integrate these outsourced applications with the internally hosted ones.

This integration problem poses many challenges beyond the basic question of getting data moving between the source and destination. SaaS applications live outside the mostly hardened corporate firewall. Security and reliability are important considerations when moving things like identifying employee, supplier and customer information (such as bank accounts, tax id numbers, etc.). These are often financial transactions so they need to be audited and have a confirmed record of completion. And finally, SaaS applications get introduced to companies usually through a departmental initiative rather than a enterprise procurement process. So integration needs to be effectively managed to not drag down the SaaS deployments while at the same time not creating a "wild west" of one off integrations across major enterprise data systems. SOA allows building architectures for integrating internally hosted IT infrastructure with SaaS, making it an important element for SaaS improvement.

### 8.1.2 SOA design principles

- SOA is an architectural style of building software applications that promotes loose coupling between components so that they can be reused. Thus, it is a new way of building applications with the following characteristics:
- Services are software components that have published contracts/interfaces; these contracts are platform-, language-, and operating-system-independent.

XML and the Simple Object Access Protocol (SOAP) are the enabling technologies for SOA, since they are platform-independent standards.

- Consumers can dynamically discover services. Services are interoperable.
- The figure below gives an overview diagram of service-oriented architecture.

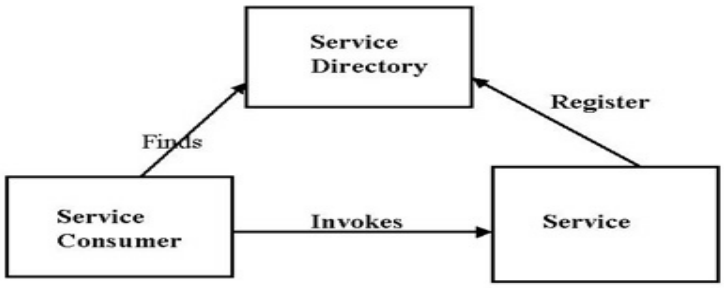


Fig 1. Service Oriented Architecture

**Figure 29: Service Oriented Architecture**

- The basic building block of SOA is the *service*. A service is a self-contained software module that performs a predetermined task. Services are software components that do not require developers to use a specific underlying technology. As Java developers, we tend to focus on reusing code; thus, we tend to tightly integrate the logic of objects or components within an application. However, SOA promotes application assembly because services can be reused by numerous consumers.
- The other key advantage of SOA is that it lets automate business-process management. Business processes may consume and orchestrate these services to achieve the desired functionality. Thus, new business processes can be constructed by using existing services.

#### *8.1.2.1 (Reference, 2006)Definition of a Service*

The definition of a service proposed in this paragraph is given by OASIS in [Oas 06]. This definition conforms to the one given by Thomas ERAL [Era 07] ias well as by the CDBI forum[8]

1. A **service** is a mechanism to enable access to one or more capabilities, where the access is provided using a prescribed interface and is exercised consistent with constraints and policies as specified by the service description. A service is provided by an entity – the **service provider** –for use by others, but the eventual consumers of the service may not be known to the service provider and may demonstrate uses of the service beyond the scope originally conceived by the provider.
2. A service is accessed by means of a service interface, where the interface comprises the specifics of how to access the underlying capabilities. There

[8] The CBDI Forum is a U.K. firm focused on accelerating the successful adoption of Service Oriented Architecture best practices through independent consulting, education and a global knowledgebase. It is considered as a reference in the area of SOA.

> are no constraints on what constitutes the underlying capability or how access is implemented by the service provider. Thus, the service could carry out its described functionality through one or more automated and/or manual processes which themselves could invoke other available services.

A service is opaque in that its implementation is typically hidden from the **service consumer** except for (1) the information and behavior models exposed through the service interface and (2) the information required by service consumers to determine whether a given service is appropriate for their needs.
The consequence of invoking a service is a realization of one or more real world effects. These effects may include:

Information returned in response to a request for that information, a change to the shared state of defined entities, or some combination of (1) and (2).

> **Note :** the service consumer in (1) does not typically know how the information is generated, e.g. whether it is extracted from a database or generated dynamically; in (2), it does not typically know how the state change is affected.

The service concept above emphasizes a distinction between a capability that represents some functionality created to address a need and the point of access where that capability is brought to bear in the context of SOA. It is assumed that capabilities exist outside of SOA. In actual use, maintaining this distinction may not be critical (i.e. the service may be talked about in terms of being the capability) but the separation is pertinent in terms of a clear expression of the nature of SOA and the value it provides.

#### *8.1.2.2 Example of service*

- In the enterprise IT area, a service could be an invoice management service, able to provide:
  - the status of the invoice,
  - its total amount, and
  - the amount of taxes.
- In the medial system, a service could be the management of image processing operations, delivering the ability to configure the image rendering: frequency, quality, outputs, etc.

#### *8.1.2.3 Ideal services*

In the context of SOA an “ideal” service has the following characteristics:

**Table 10: characteristics of an ideal service**

| Characteristic | Description |
|---|---|
| Services represent a business domain | The real power of an SOA lies in its ability to model a business domain. A method based on a transverse process model allows getting in output a list of functional services. To each functional service will be associated an application service that belongs to an application either developed or based on COTS. To be executed, an application service needs technical services. The application of SoC principle which allows integrating technical services with functional services is extended with ESB to application modules i.e. application services. |
| Services have a modular design | Services are composed of modules. A module can be thought of as a software subunit or sub system that performs a specific, well-defined function. A module is a set of software units that provides value to a service consumer. |
| Services are loosely coupled | Loose coupling means that there are no static, compile-time dependencies between clients and providers. Consumers dynamically find providers at run-time. |
| Services are discoverable and support introspection | Dynamic discovery and binding are a key point for flexibility and reusability. Technologies such as OSGi and registries are a way to support this characteristic. |
| Location of service is independent to client | Making the location of service transparent to the client allows tremendous flexibility in the implementation of a SOA. So the service can be located wherever is most convenient. Instances of the service can be spread across several machines in a cluster. Hosting of the service can be moved to a third-party provider. |
| Platform independent | For the greatest flexibility, a service should not care upon which platform it is running. Therefore the ideal service should theoretically be independent from the operating system hosting it. For instance, a service implemented on a virtual machine is independent from the hosting platform. |

#### *8.1.2.4 Service oriented programming*

- Service oriented programming considers that each service that respects a service contract can be substituted to another one following the same contract. Besides, choosing a candidate among a list of services is decided as late as possible at execution time. The client gets a list of services from a registry. A provider registers its services in a registry with an associated qualified contract. Dynamic service oriented programming considers also that services useful to a client can appear at any time and that currently used services can disappear at any time. So it is necessary that the client is aware of these changes. This type of programming is used for Web Service and OSGi.

#### *8.1.2.5 Service network*

The goal of a service network is to provide an infrastructure for next-generation applications. Service network is based on the Service Component Architecture (SCA) concepts that provide:

- A way to represent a service network, termed an *assembly,*
- A model for constructing components in a variety of languages including Java, C++ and BPEL,
- Also defines integration with existing programming models including Spring and EJB.
- A model for associating policies with services.

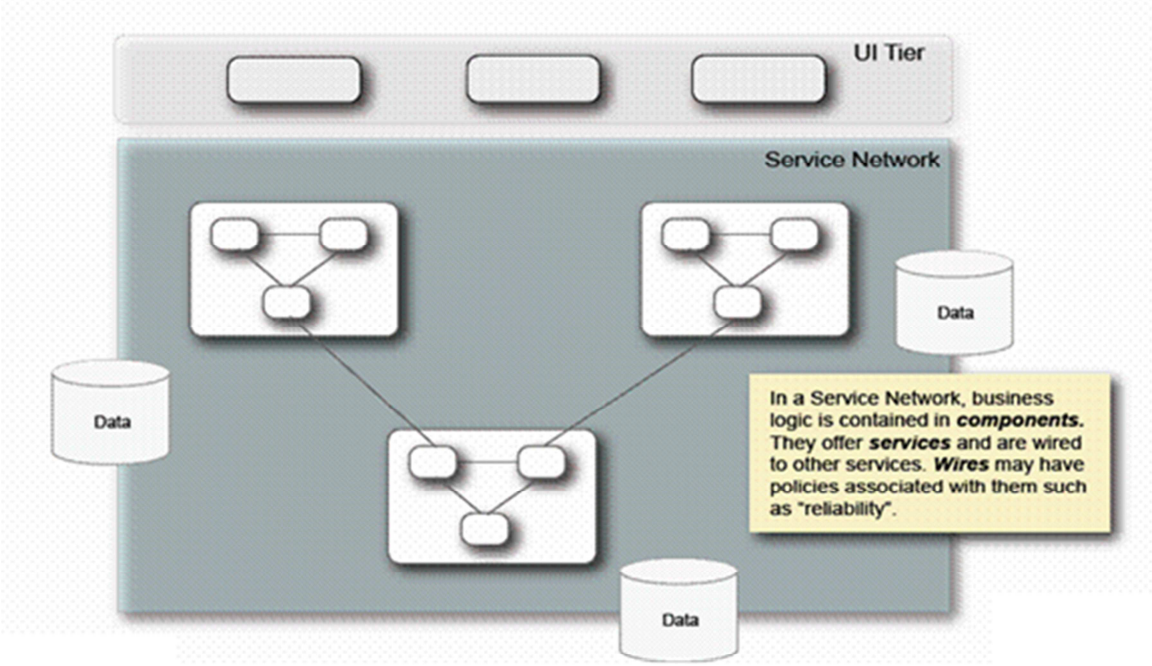


**Figure 30: the service network**

#### *8.1.2.6 The mediation pattern*

- The mediation pattern is the principle according which service consumers and service providers should never directly communicate. A mediation element, acting as a proxy, should be inserted between service consumers and service providers for managing exchanges between them. Typically, An ESB could play the role of mediator.
- This is a key element for loose coupling, and consequently for the openness and the flexibility of the IT system, its ability to evolve and accept new services and/or replace services by others equivalents. This principle is considered as a founding principle for SOA.

- The article in the frame below is extracted from a ZapFlash from ZapThink (see [Sch 08]) and matches the vision of Bull on the need for a mediation pattern in an SOA.

Many people think they are doing or talking about Service-Oriented Architecture (SOA), but most of the time they're really doing Web Services Integration (WSI), developing Just a Bunch of Web Services (JBOWS), or even just point-to-point XML (what many would like to call REST, but even in most cases is not even truly Representational State Transfer [REST], which is also an architectural form).
SOA is an architectural concept; SOA is not a new name for integration.
**The Problem of Service Consumer to Service Provider Communication**
In typical integration environments, you're primarily concerned with the movement of data or the amalgamation of business logic to achieve a specific task. In that vein, most integration developers think about inputs ("gazintas"), outputs ("gazattas"), transfer, and transformation. That is, they think about how they're going to get data or application functionality out of one system, how they'll get it into another system, the mechanism they'll use to get that information or functionality from one place to another, and then what they need to do with it while it's en-route to transform it from one format, protocol, context, process, or policy into another. So, while architects are chemists – that is, scientists who are trying to understand a system and how it can all be put together, integration developers are chemical engineers – focusing on just the mechanisms of transformation.
From this integration-centric mentality, most integration developers see the concept of the Service as just another means of gazinta or gazatta. That means these integration-heads need to find something else to do transfer and transformation. Most vendors responded to this desire by simply dusting off their existing integration middleware. End-users responded easily to this insignificant change and voila, now they have transfer and transform that can deal with their Service-based gazintas/gazattas. But does this make SOA? No.
The reason why this approach is wrong is because Services Consumers should ***never*** communicate directly to Service Providers.
Why? First, the main concept of SOA is that we want to deal with frequent and unpredictable change by constructing an architectural model, discipline, and abstraction that loosely-couple the providers of capability from the consumers of capability in an environment of continuous heterogeneity. This means that we have to somehow develop capabilities without the knowledge of how they might be used. This means that we need to have significant variability at a variety of aspects including the implementation, infrastructure, contract, process, policy, data schema, and semantic aspects. Having this variability allows companies to have stability in their business and IT even though the IT and business continue to change. Agility, predictability, reliability, visibility, and cost-effectiveness all become that much more realistic when we can achieve that sort of abstraction.
But to achieve that abstraction means that we can't afford things to break when things change. Lots of things change in an IT or business environment: location and availability of services, versioning of Service implementations and contracts, changes to business processes and policies, new or changing requirements for reporting, visibility, and control, and of course the need for new capabilities to solve emerging business problems or capitalize on new business opportunities. However, if you think that a Service is just an API,

then all hell breaks loose when you try to connect a Service Consumer directly to a Service Provider. What if the provider moves? What if the Service contract changes? What if the provider is no longer available? What if you now need a new communication mechanism or change to a data schema?
The first knee-jerk reaction to this complexity by most integration "architects" is to put some piece of technology between the Service consumers and providers to ameliorate this problem. The impulse is correct: you can't have direct communication if you want to have variability. However, the execution is all wrong. If you are relying on black-box, and often proprietary, technology to do the heavy lifting of managing Service communication differences you'll simply shift all the complexity and work from the end-points (the Services) to an increasingly more complex, expensive, and brittle middle point. Welcome to EAI 2.0, the secret and unpublicized real name for most so-called Enterprise Service Bus (ESB) products – or at least the way that many are using them.

**(Re)Introducing the Service Proxy**

So, if reworked middleware isn't the solution, what is? Well, of course the answer is architecture. But that's too pat of an answer, so to speak again in the parlance of integration developers, let's explore how a system would be designed if you can't count on any particular vendor's solution to be in the middle to make it all work.
If a Service Consumer wants to consume or compose some functionality, but the system does not know functionality is or even, how to communicate with it. There are at least three ways of handling this particular challenge. First, talking to a proxy (or broker or intermediary) in a language that you do understand and that's in a location that you already know and send a request to this proxy that will act on the behalf of the Consumer. This is the "envelope within an envelope" concept. Basically, consider a message to be delivered to someone that you know, but that you do not know where she or he is located, within an envelope that is addressed to a party that you do know. That party then opens the envelope and addresses the internal envelope to the end recipient based on a look-up of some sort. This approach simplifies the problem of the Service consumer having to know about changes since they just need to know about the proxy and the end-recipient. From a technology perspective, the use of WS-Addressing simplifies this approach. In essence, a Service consumer just needs to know the WS-Address of a Provider and hand that to a Service proxy to resolve and deliver.
The problem with this approach is that the Service proxy still needs to find a way to deliver the message, and if the proxy contains all the rules internally in the black-box, we have the same problem we have with EAI 2.0-style ESB solutions. This is where the registry steps in. Service proxies should not store their own rules, but rather get all their routing, location, and policy-based binding rules from a central system of record. In SOA, this is the role of the registry. Proxies can cache the results of registry look-ups for performance enhancement, but the metadata themselves should come from the registry.
So, registry-configured Service proxies solve many of the problems of Service communication, but it still leaves open the problem of the connection between the Service consumer and the Service proxy. What if the proxies move or are unavailable? And how do Service consumers find out about the Service proxy in the first place? The answer to this challenge is the use of registry-based "late binding". In this scenario, the Service consumer pings the registry to find out where the proxy is, perhaps based on where the Service consumer is

currently located and with what protocols it can communicate. One would think that we can make the Service consumer directly bind to a Service provider after finding its metadata in the registry, but that would be a mistake. The problem is that if the Service provider is using some alternate protocol or is in some location that the Service consumer can't reach, we'll have a communications problem. So, even in the case where we're discovering metadata in a registry, we still don't want direct communication between a Service consumer and provider.
Therefore, the optimal solution here is to use registry-based "late-binding" to help the Service consumers find proxies and also to help them resolve Service providers to WS-Addresses. Then, the Service consumer communicates with a proxy, providing the WS-Address and required metadata for communication. This allows Service consumers to be quite simple, leveraging a network of distributed, and most likely heterogeneous, Service proxies doing the communication on behalf of Service consumers. In this scenario, companies can implement Service proxies using and mixing a wide range of technologies and approaches including hardware-based network intermediaries, software proxies, and even agents that reside on Service consumer end-points.
Finally, in the above scenario, where's the ESB? Nowhere, unless you believe that the ESB is a pattern and not a product (as IBM once espoused). Is the ESB needed? Not necessarily. Will it make you feel better to buy one anyways? That may be. Of course, what are these proxies? Surely ESB vendors will want to have you believe that their products can be used as proxies in this configuration, and that's absolutely true. They certainly can. The point is not that ESB's can't make good proxies. They certainly can, and I want to specifically point that out. But you can also get away with not using one or using your existing infrastructure and it will all still work. The sad truth is that most integration architects are not using ESBs as Service proxies, but rather as EAI 2.0 hub-and-spoke or message-bus brokers that facilitate Web Services Integration, not SOA.
**The ZapThink Take**
If you think about it, we've already implemented complex, heterogeneous networks in this fashion before. Computers leverage DHCP, gateways, and routers to do networking in a way that's far superior to direct, point-to-point IP communications. And this is basically what we're saying about Service communication. Understanding this is a realization that this is not a technology issue, but an architectural one.

It should be noted that this mediation pattern is not necessarily in opposition with real-time constraints that could exist on a system. Eventually, this could imply that the mediation component involved has itself to satisfy the real-time constraints.

#### *8.1.2.7 Event driven architecture and complex event processing*

8.1.2.7.1 Understanding Events

Today's enterprises-business, government, and military are adapting to and utilizing global information processing. The enterprise is operating in a complex environment. These are high-level business, logistics, and application-to-application events. They form the global event cloud in which the open enterprise is operating. We talk of a

cloud of events rather than a stream because the event traffic is not, in most cases, nicely organized. They do not necessarily arrive in the order they were created or in their causal order.
An incoming event generated by an alarm, a signal, for instance an interrupt triggered by a sensor, the arrival of a file, or of a message, can be formalized as an event-type message. More and more, events take place in system architectures. These architectures are named EDA (Event Driven Architectures). Events coming from several sources are analyzed by CEP (Complex Event Processing) software and events in general trigger a process.
The messages arrive in a queue where they are stored until they are processed. Queues managements are implemented with different mechanisms. MOMs[9] are an example of such mechanism implemented in message buses.
The event technology provides the following:

- An ability to design and deploy event-driven parallel and asynchronous processes that can recognize complex event situations in the global event cloud
- Process simulation as part of the design phase, before going live
- On-the-fly modifiable processes
- Design for handling exceptional situations as an integral part of process design
- Real-time, drill-down, event-based diagnostics, utilizing event causality relationships, applicable to process behavior

CEP defines precisely what an event is[10]. In everyday usage an “event” is something that happens. In CEP an “event” is an object that can be subjected to computer processing. It signifies, or is a record of an activity that has happened. *An event is an object that is a record of an activity in a system*. The event signifies the activity. An event may be related to other events.
An event has three aspects:

- *Form*: The form of an event is an object.
- Significance: an event *signifies* an activity. We call the activity the *significance* of the event. The form of an event usually contains data describing the activity, its significance and relativities.
- Relativity: An activity is related to other activities by time, causality, and aggregation. Events have the same relationships to one another as the activities they signify. The relationships between and event and other events are together called its relativity.

#### 8.1.2.7.2 Layered Architectures and Plug-and-Play

Layering is a technique for controlling complexity.

A process at one level in architecture is represented by behavior rules and constraints. That is the highest-level view of that process, called the process interface or class. It

[9] For: "Message Oriented Middleware"
[10] The definition of an event, given in this section comes from:”The Power of Events” by David LUCKHAM.

tells us what the process does – and what it must not do. When we ask for « more detail » as to how the process carries out those rules, we go down a level in the architecture to a lower level, more detailed view. This is like "opening the interface" and looking inside the process. What is seen is called the implementation of the process.
In a layered architecture, an implementation can be a lower-level architecture of processes. In this case, the lower-level architecture implements the process interface. The lower-level processes are called sub processes of that process.

#### 8.1.2.7.3 Abstraction Principle

The abstraction principle governs the relationship between architectures of sub-processes and the process interfaces that they implement.
Architectures of sub-processes that implement a process interface must obey two rules:

- Interface conformance: the architecture of sub processes must conform to the behavior rules and constraints in the process interface.
- Communication conformance: Sub processes can communicate with processes outside their sub-architecture only by receiving input events or creating output events that are defined in the process's interface.

***Interface conformance*** means that we should not be able to distinguish by monitoring the runtime behavior at the interface, consisting of input and output events at the interface of the process, whether the behavior rules in the interface are being executed or sub-processes are being executed. Also, design constraints in the interface should not be violated.
***Communication conformance*** means that all communication between sub-processes in different boxes is through the interfaces of those boxes.
Typical object-oriented programming language such as Java do not enforce the second rule of the abstraction principle but allow a class that implement an interface to make method calls to other interfaces at the same level.
Any sub-process in an implementation communicates with processes outside of its sub-architecture through the interface of the process it implements. This gives us two important capabilities.

- Simulation before implementation: The ability to simulate the processes and their communications using only the behavior rules specified in the interfaces of the processes and connectors at that level.
- Plug-and-Play with reduced risk of errors: The two rules of the abstraction principle ensure that any two implementations for the same process interface conform to the rules and constraints in the interface.

#### 8.1.2.7.4 Event Driven Architectures (EDA)

**Event-driven architecture** (**EDA**) is a software architecture pattern promoting the production, detection, consumption of, and reaction to events.
Event-driven architecture complements service-oriented architecture (SOA) because services can be started by triggers such as events. The article in the frame below is extracted from a ZapFlash from ZapThink (see [Mar 12]) and matches the vision of Bull on this topic.

EDA is an approach where events trigger asynchronous messages that are then sent between independent software components that are completely unaware of each other – in other words, decoupled, autonomous objects. An event source typically sends messages through some middleware integration mechanism like a bus, and then the middleware publishes the messages to the objects that have subscribed to the events. The event itself encapsulates an activity, and is a complete description of a specific action.
SOA, on the other hand, is an architectural approach where software functionality is exposed as loosely coupled, location independent Services on the network.
Are SOAs Event-Driven?
To understand why EDA is a subset of SOA, we need to get into the details of the underlying standards, beginning with SOAP. SOAP supports four interaction patterns: request-response (client to server and back), notification-response (server to client and back), one-way from client to server, and notification from server to client. In this way, SOAP by itself is able to support both remote procedure call (RPC) and document-style interactions, in either a synchronous or asynchronous fashion. In particular, the event notifications cores to EDA have been a part of SOAP from the beginning.
SOAP by itself, however, does not provide all the detail needed for a full Event-Driven publish/subscribe environment. As a result, there are several initiatives working their way through vendor groups and standards bodies meant to complete this picture: one camp is working on Web Services Notification (WS-Notification), Web Services Base Notification (WS-BaseNotification), Web Services Brokered Notification (WS-BrokeredNotification) and Web Services Topics (WS-Topics), while another camp is working on Web Services Eventing (WS-Eventing), Web Services Dynamic Discovery (WS-Discovery), Web Services Coordination (WS-Coordination), Web Services Metadata Exchange (WS-MetadataExchange), and Web Services Business Activity Framework (WS-BusinessActivity). Now, it's obvious that much work remains to bring this laundry list of specifications into a single streamlined set of standards, but be that as it may, once the dust settles, SOA based on Web Services will offer a complete, robust set of Event-Driven mechanisms.
More importantly, ZapThink discusses frequently in its research that as application logic and functionality becomes more coarse grained, they become more “business-like” and less “API-like.” At the highest level, all of the details of how the company operates (its processes and services) are hidden from the Service consumer. Invoking these Services then becomes a matter of sending the right messages (or events) to trigger processes to occur that in turn generate subsequent events for further Service processing.
Are all EDAs Service-Oriented?
If you've been reading this article closely up to this point, you will have noticed a subtle, but critical difference between SOAs and EDAs: SOAs should be loosely coupled, while EDAs can be decoupled to a much greater degree. In other words, in an SOA, Web Service producers and consumers need have no knowledge of the workings of the other except what is expressed in the Service contract (i.e., the WSDL file for the Service), while in an EDA, there is no need for any such contract. The only connection between event producers and consumers are the "publish and subscribe" activities themselves.

The big problem with fully decoupled EDAs, however, is that they are not particularly good for application-to-application communication. This level of decoupling works for completely unstructured information, for example, textual information intended for human consumption. When one application publishes data for another application to consume, however, in the absence of a Service contract, those data are necessarily fine-grained. In other words, a fully decoupled event might be an alert that a process is complete or might be some kind of acknowledgment, but the subscribing application would be hard pressed to make sense out of a complex, structured event in the absence of a Service contract.
The bottom line, then, is that it's possible for an EDA not to be Service-oriented, but for most practical purposes, Event-Driven interactions in an EDA should be Service-oriented. The distinction between the two approaches, therefore, is more of a technical detail than a difference that has any importance to the business. So, what we should be talking about is not a separate concept called EDA, but rather "event-driven SOAs" as a coherent melding of the two concepts.

#### 8.1.2.7.5 Complex Event Processing (CEP)

Complex event processing software is the engine behind a new generation of real-time applications which allow finding and taking advantage of new opportunities in rapidly changing electronic markets, understand customer behavior in real-time to increase wallet-share or always knowing where products and assets are allocated and for realignment on a moment's notice. In effect, CEP makes the business more agile. Complex event processing software is a new breed of off-the-shelf infrastructure software that can be applied to a large number of business and technology solutions across multiple industries. Scalable, reusable, off-the-shelf CEP software:

- Allows real-time, event-driven applications to be developed faster and at a lower cost
- Reduces long-term maintenance costs of real-time applications
- Drives an economy of scale cost benefit for programming and administration

### 8.1.3 SaaS design principles

#### 8.1.3.1 SaaS revisited

Software-as-a-Service (SaaS) shares the distinction of being both a business model and an application delivery model. SaaS enables customers to utilize an application on a pay-as-you-go basis and eliminates the need to install and run the application on the customer's own hardware.

Customers generally access the application via a Web browser or thin client over the Internet. SaaS is most often subscription-based and all on-going support, maintenance, and upgrades are provided by the software vendor as part of the service. Application customization capabilities, if available at all, are generally provided to all customers in a consistent manner. From the perspective of the software vendor, the SaaS model provides stronger protection of its intellectual property, operational control of the environment running the software, and generally a

repeatable revenue stream from the service subscription fees. Software vendors have varying capabilities and applications can come in varying manners but SaaS applications most typically support many unique customers using a single instance of that application, what is also known as multi-tenancy.

The figure below positions pattern of SaaS with usual models of deployed services.

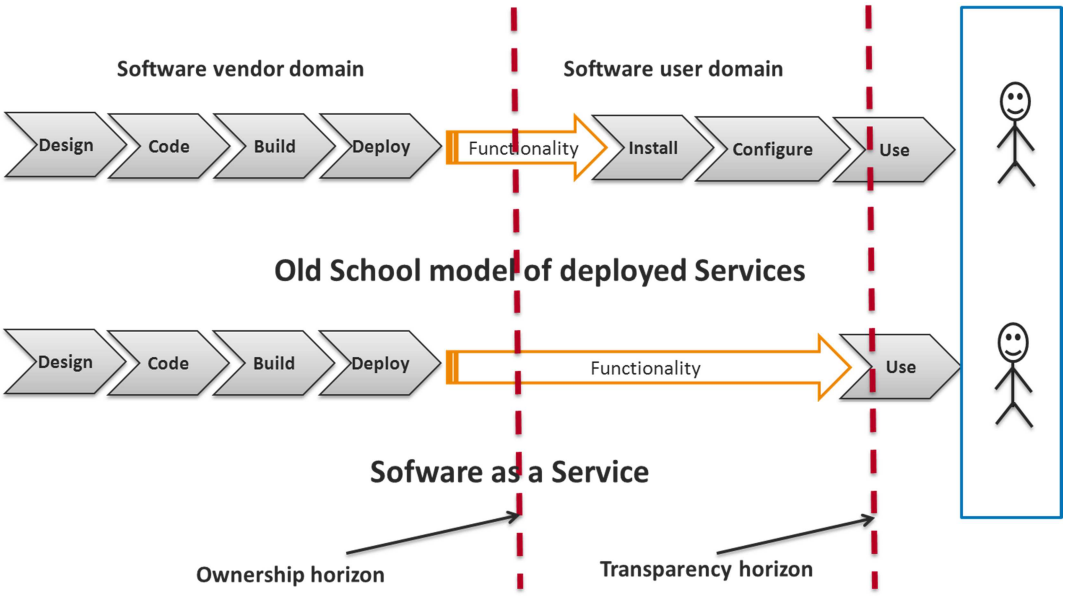


**Figure 31: pattern of SaaS - different domains of ownership**

### 8.1.3.2 Usual SaaS information system architecture: conceptual view

The conceptual architecture is intended to represent a three- to five-year vision of SaaS architecture, free of implementation technology details, and to establish common capability definitions. The conceptual architecture depicts all the key capabilities required in a complete SaaS offering, the logical separation of capabilities into tiers, and the logical grouping of capabilities. It's not expected that individual SaaS applications will necessarily include every capability described in the conceptual architecture. The figure below summarizes this conceptual view.

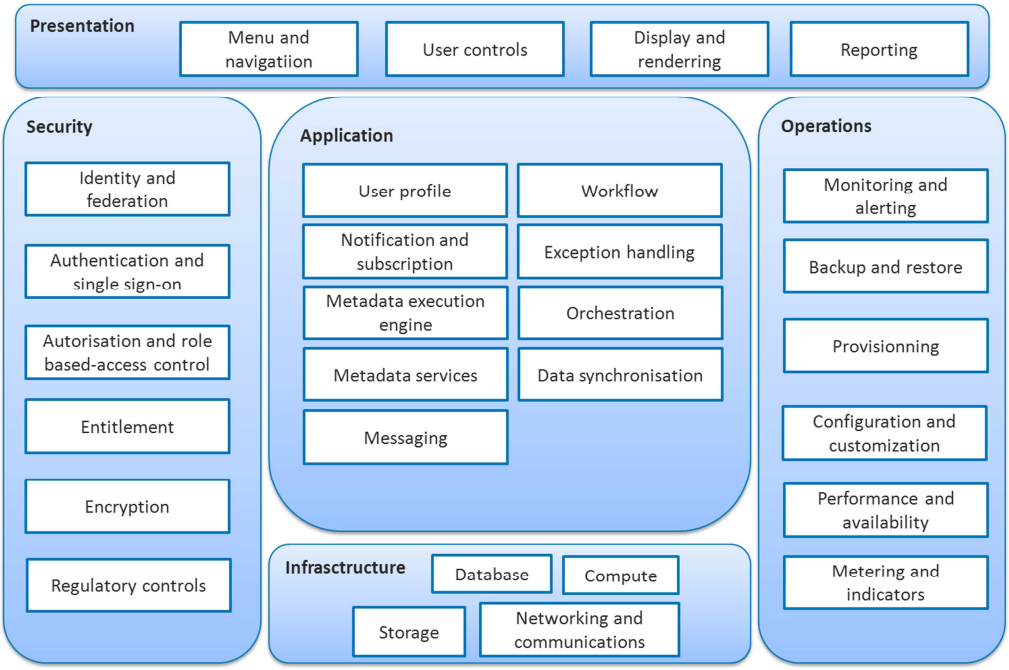


**Figure 32: SaaS Information System Architecture - conceptual view**

#### *8.1.3.3 Key features*

A well-designed SaaS application has several key architectural features. It should be:

- **Multi-tenant efficient**. The design should support multiple tenants using a single instance of the application. The data must be segregated for each tenant.
- **Configurable**. The application can be configured to meet the needs of each tenant, using metadata and a metadata execution engine—also known as a business rules engine. Routine configuration changes should be possible without the need to coordinate downtime with other tenants.
- **Scalable**. Multi-tenant usage can result in millions of users. Applications should be designed from the ground up to scale up and scale out—and to be able to do this dynamically, on demand.

## Operations

These are the capabilities needed to efficiently keep the SaaS application running:

- **Monitoring and alerting**. Polling application components, services, and infrastructure to detect failures. On detection, an alert is sent to the appropriate support group.
- **Performance and availability**. Performance describes how the application performs under load, both in terms of the number of users and the transaction volume. In the context of SaaS, this should allow applications to dynamically scale based on runtime usage and demand. Availability is a measure of how much of the time the application is available to users and is represented as a percentage.
- **Metering and indicators**. Tracking and reporting items specifically related to the service-level agreement, such as usage, availability, number of failures, and mean time to respond to and fix problems.

#### *8.1.3.4 SaaS capabilities*

Many capabilities make up the SaaS conceptual architecture. We group these into presentation, security, application, operations, and infrastructure categories, as shown in Figure 32. The following sections describe the most important capabilities.

### Presentation

This includes all capabilities exposed to the user, such as:

- **Menu and navigation**. These provide access to the features and functionality within an application, organized in an intuitive way so that the user can select the desired function.
- **Reporting**. Application-specific predefined or ad-hoc reports.

### Security

Security is one of the most important categories of SaaS capabilities, given that Intel's data and user accounts are typically hosted by the SaaS provider. We considered the following capabilities:

- **Identity and federation**. Identity uniquely identifies a user or another entity such as an Intel application or system. An example is a user name. Federation describes the function of enabling users in one domain to securely and seamlessly access data within another domain.
- **Authentication and single sign-on (SSO)**. The process of identifying an individual usually based on a user name and password. In the context of SaaS, this includes the ability to achieve SSO across multiple cloud applications and services.
- **Authorization and role-based access control**. After an identity has been confirmed, authorization is the process of giving individuals access to system objects based on their identities. Identities are usually assigned to roles for ease of managing access.
- **Entitlement**. The process of granting access to a specific resource. Tenants are usually responsible for maintaining their own user accounts using delegated administration.
- **Encryption**. Data may need to be encrypted in transit (between applications or between the layers within an application) and at rest (while stored).
- **Regulatory controls**. Tracking and reporting who accessed what, when, and why. It includes tracking access to application features and data, the security rating of the data, and the implementation of a data retention policy. It also includes identifying whether individuals are located in controlled countries.

### Application

These represent the typical business layer or middle tier of a SaaS application:

- **User profile**. The attributes and information that describe a user, such as name, e-mail address, and role.
- **Metadata execution engine**. Statements that define or constrain some aspect of the business. They are intended to assert business structure or to control or influence the behavior of the business.
- **Metadata services**. Information about which data is contained and exposed within an application and about how content is organized.

- **Workflow**. The defined series of user-based tasks within a process to produce a final outcome. An example is creating a purchase order.
- **Exception handling**. The process of raising and managing exceptions within an application. This includes how application errors are exposed to the user and how error messages are logged.
- **Orchestration**. The series of technical tasks performed within a process to produce a final outcome. An example is an extract, transform, and load sequence to move data between business applications.
- **Data synchronization**. The capabilities for synchronizing data held within the application with external data.

**Infrastructure**

The underlying technical capabilities required for storing data and moving it around the network:

- **Database**. In multi-tenant data architecture, there could be one database per tenant or one database shared by multiple tenants with the data indexed by a specific tenant identification.
- **Compute**. The physical clients, servers, or virtual machines that execute code.

### 8.1.4 SOA and SaaS technologies

#### *8.1.4.1 OSGi: the dynamic module system*

8.1.4.1.1 Introduction to OSGi Solution and Its Architecture

The OSGi Service Platform provides the functions to change the composition dynamically on the device of a variety of networks, without requiring restarts. To minimize the coupling, as well as make these couplings managed, the OSGi technology provides a service-oriented architecture that enables these components to dynamically discover each other for collaboration.
OSGi provides a component-oriented, lightweight container framework for hosting dynamically managed services. A module, called bundles in the OSGi, is a jar containing interfaces, their implementations, other classes, and a manifest. A bundle is a basic module for packaging, deploying, and managing the lifecycle of a group of services.

- *OSGi service*

An OSGi service is a Java object instance, registered into an OSGi framework with a set of properties. Any Java object can be registered as a service, but typically it implements a well-known interface.

- *OSGi Bundles Provide an Isolation Model*

Each OSGi bundle encapsulates packaged services and controls their visibility through metadata in a manifest file. As shown in Figure 33, a package or service in a module can be made available to other modules by explicitly exporting it.

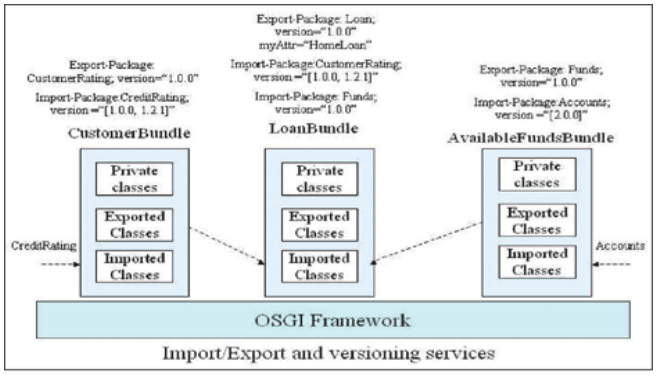


**Figure 33: OSGi bundles providing encapsulations, dependencies and versioning**

Other modules can't access any internal, non-exported interfaces or their implementations or other internal classes. In Figure 38, the import statements are to be noticed. Classes from other bundles can use these exported services by explicitly declaring imports.

- *OSGi Container Manages Lifecycle & Supports Dynamic Loading*

Modules go through lifecycle events such as deploy (install), start, stop, redeploy, and uninstall. When a module (bundle) is being installed, the container checks all the dependencies (imports). If all the dependencies are satisfied, the module is ready to start. Figure 34 shows the lifecycle management of OSGi bundles. The OSGi runtime container provides a lightweight microkernel in which modules can be dynamically added or removed.

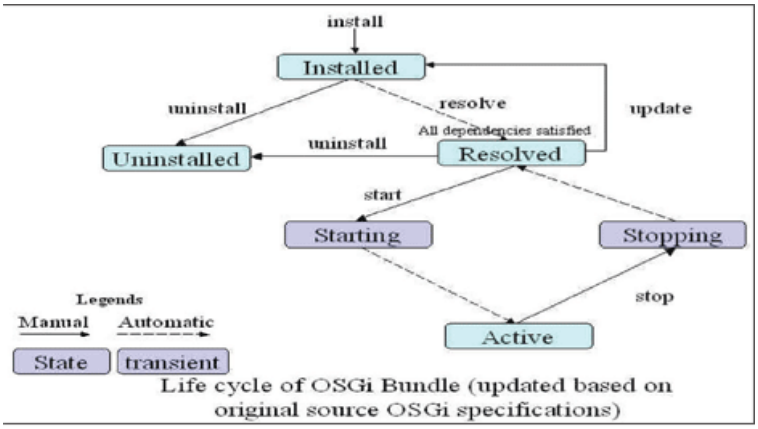


**Figure 34: OSGi lifecycle management**

- *OSGi Natively Supports Dynamic Versioning of Services*

During the long life span of enterprise software, business and technology changes. The software must reflect these changes and upgrade different modules independently of each other. For example, a company might have a customer data model (CDM) for its sales, order capture, and marketing systems. When there's an upgrade of the CDM, the company might choose to upgrade sales and marketing systems. However, upgrading the order capture system could have implications for business process or stability and so might not be upgraded at the same time, resulting in a need to support two versions of the CDM simultaneously.
The OSGi specification natively supports versioning with version attributes in the export and import instructions (see **Figure 35**) in the manifest files. It also provides

additional control via arbitrary export/import attributes. The selection of an imported service or bundle is based on its version and attribute(s). OSGi also provides a unique isolation model in which multiple versions of the same service can co-exist and each service instance is isolated from the potential problems of class conflicts. A service consumer can get a reference to the correct version by specifying a version and other attributes in a filter as shown in the next figure.

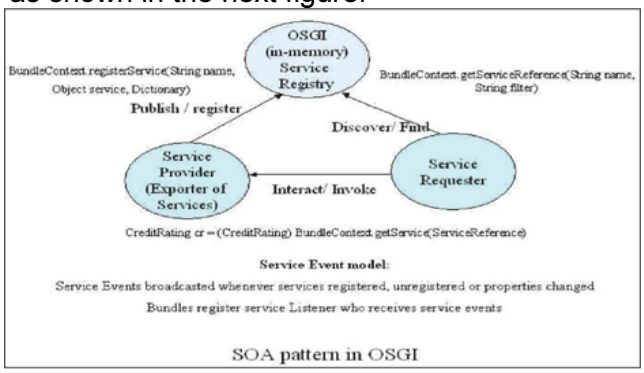


**Figure 35: SOA pattern and the dynamic lifecycle of services**

- *OSGi Provides a Service-Oriented Programming Model*

OSGi also provides a programming model based on SOA. It is important to note that OSGi natively provides service-oriented programming in a dynamic environment. Unlike the RMI service registry, the OSGi service registry is native to the container - the services are automatically registered during the load without having to register them programmatically. A publisher can register a service using the *registerService* method in its *BundleContext*. Similarly, a service requester can discover a service using the *getServiceReference* method from its *BundleContext*. Unlike EJB or RMI, the services don't need to implement any remotable interface. It's all POJO!

- *OSGi Can Handle the Dynamic Lifecycle of Services*

Again, in enterprise software, modules can be added or removed. Since the container automatically manages the registration and deregistration, application programmers have to understand that services come and go at any time. OSGi provides a framework to track and react to lifecycle changes in the service, sending notifications of the registrations and de-registrations to a service layer that also provides a service tracker utility to help client programming in this dynamic environment. There is also a declarative model to handle lifecycle changes. Other containers like EJB/RMI don't handle such changes in lifecycle.

#### 8.1.4.1.2 OSGi Architecture & Services

The OSGi platform provides core framework and platform services. The core framework provides a runtime foundation to run and manage the lifecycle of various applications in a secured and modularized environment. The framework has four layers: security, modularization, lifecycle, and service.
The security layer provides additions to Java security, restricting the public and private exposure of packages and services and also provides permissions to import/export packages and register/access services from the service registry. The modularization

layer handles class loading, versioning, and the import/export of packages. It also manages the dependency resolutions of bundles.
The lifecycle layer manages the lifecycle of bundles and provides a generic API abstraction to enable remote management in a variety of management models.
The service layer adds a dynamic behavior to the OSGi platform in which bundles can come and go. The heart of this layer is a service registry in which services are registered and discovered. The framework handles automatic registration and deregistration and triggers lifecycle events. OSGi further provides a declarative model to express service registrations and dependencies in an XML declaration. This declarative framework supports lazy (delayed) loading of resources by loading them only when they're actually needed.
The OSGi ecosystem platform provides various service interfaces that can be implemented by vendors, depending on the nature of their applications. The OSGi framework may provide a permission admin service, conditional permission admin service, a package admin service, and URL handler service. The OSGi Alliance specifies various system services such as a log service, and administrative services for managing configuration, event administration, users, devices, and applications. The Alliance has also defined an HTTP service so that bundles can provide servlets that can be made available over HTTP. Besides declaration and event services, in release 4, the Alliance defined a wire admin service to manage configuration-based connectivity between a service producer and consumers, enabling data objects to be exchanged over a wire (see next figure).

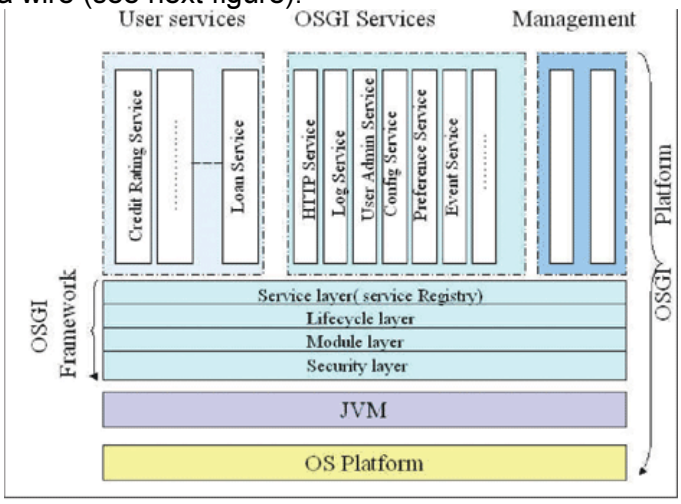


**Figure 36: The OSGi Framework and Platform**

### *8.1.4.2 Web Services*

Web services is a technology for transmitting data over the Internet and allowing programmatic access to that data using standard Internet protocols. The term Web service is not used to represent a company who simply offers services on the Web, such as a banking Web site. Although such a company offers a service over the Web, it doesn't necessarily make its service available by using a programmatic interface that allows two applications to be integrated. In fact, a Web service allows a developer to include functionality into a program without needing to "reinvent the wheel" and without needing to know anything about the business or complexity of the Web service that he or she is using.

For example, let's suppose keeping a database up-to-date with information about weather in Gabon. In order to distribute that information to anyone in the world. To do so, the weather information could be published through a Web Service that, given a ZIP code, will provide the weather information for that ZIP code.

The clients (programs that want to access the weather information) would then contact the Web Service (in the server), and send a service request asking for the weather information. The server would return the forecast through a service response. The figure below is a very sketchy example of how a Web Service works.

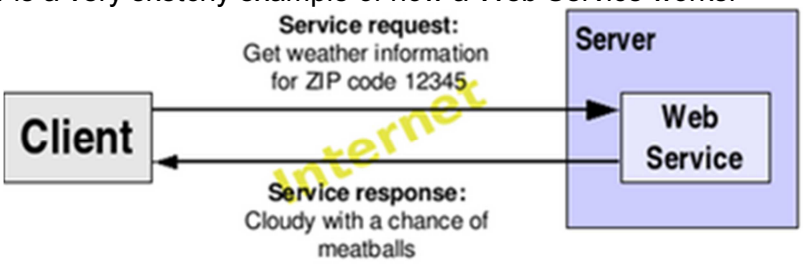


**Figure 37: Web services invocation - simple view**

Here is how it actually works.

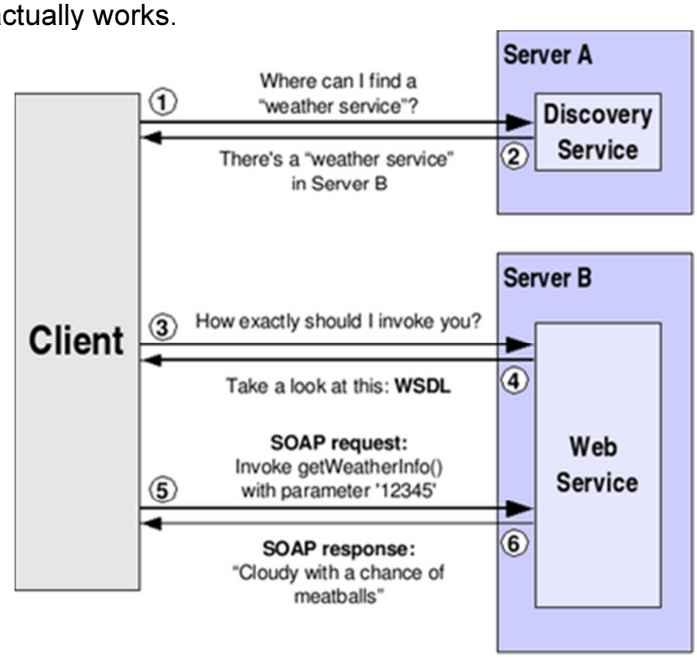


**Figure 38: Web services invocation - detailed view**

#### 8.1.4.2.1 Web services architecture

SOAP and WSDL are essential parts of the Web Services Architecture detailed below.


**Figure 39: Web services architecture - functional view**

The figure above shows the functional view of usual architecture for web services.
Here is the description of each layer of this architecture:

- **Service Processes**: This part of the architecture generally involves more than one Web service. For example, discovery belongs in this part of the architecture, since it allows us to locate one particular service from among a collection of Web services.
- **Service Description**: One of the most interesting features of Web Services is that they are *self-describing.* This means that, once you've located a Web Service, you can ask it to 'describe itself' and tell you what operations it supports and how to invoke it. This is handled by the Web Services Description Language (WSDL).
- **Service Invocation**: Invoking a Web Service (and, in general, any kind of distributed service such as a CORBA object or an Enterprise Java Bean) involves passing messages between the client and the server. SOAP (Simple Object Access Protocol) specifies how we should format requests to the server, and how the server should format its responses. In theory, we could use other service invocation languages (such as XML-RPC, or even some *ad hoc* XML language). However, SOAP is by far the most popular choice for Web Services.
- **Transport**: Finally, all these messages must be transmitted somehow between the server and the client. The protocol of choice for this part of the architecture is HTTP (HyperText Transfer Protocol), the same protocol used to access conventional web pages on the Internet. Again, in theory we could be able to use other protocols, but HTTP is currently the most used one.

Now, let's take a close look at what the server looks like, especially what software is expected to have to get Web services up and running on a server.

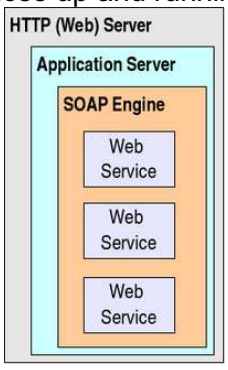


**Figure 40: The server side**

- **Web service**: First and foremost, this is basically a piece of software that exposes a set of operations. For example, if the Web service is implemented in

Java, then the service will be a Java class (and the operations will be implemented as Java methods). Obviously, a set of clients must be able to invoke those operations. However, the Web service implementation knows nothing about how to interpret SOAP requests and how to create SOAP responses. That's why a soap engine is needed.

- **SOAP engine**: This is a piece of software that knows how to handle SOAP requests and responses. In practice, it is more common to use a generic SOAP engine than to actually generate server stubs for each individual Web service (note, however, clients stubs are still needed). However, the functionality of the SOAP engine is usually limited to manipulating SOAP. To actually function as a server that can receive requests from different clients, the SOAP engine usually runs within an application server.
- **Application server**: This is a piece of software that provides a 'living space' for applications that must be accessed by different clients. The SOAP engine runs as an application inside the application server. A good example is the JOnAS server, a Java Servlet and Java Server Pages container that is frequently used with Apache Axis and the Globus Toolkit.

  Many application servers already include some HTTP functionality, so setting Web services implies installing a SOAP engine and an application server. However, when an application server lacks HTTP functionality, then an http server is needed.

- **HTTP Server**: This is more commonly called a 'Web server'. It is a piece of software that knows how to handle HTTP messages.

#### 8.1.4.2.2 Web services in practice

Despite having a lot of protocols and languages floating around, Web Services programmers usually only have to concentrate on writing code in their favorite programming language and, in some cases, in writing WSDL. SOAP code, on the other hand, is always generated and interpreted automatically. Once the client application needs to invoke a Web Service, this invocation is delegated on a piece of software called a stub. There are plenty of tools available that will generate stubs automatically, usually based on the WSDL description of the Web Service.

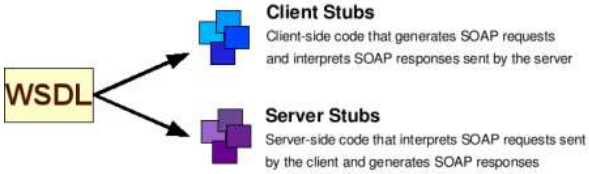


**Figure 41: Client and server stubs**

Having said that, let's describe a typical web services invocation.
Let's suppose that the Web Service to use is already located (through discovery service, or using a static URI), and the client stubs are already generated from the WSDL description. What exactly happens then when the Web service operation is invoked from a program?

1. Whenever the client application needs to invoke the Web Service, it will really call the client stub. The client stub will turn this 'local invocation' into a proper SOAP request. This is often called the *marshaling* or *serializing* process.
2. The SOAP request is sent over a network using the HTTP protocol. The server receives the SOAP requests and hands it to the server stub. The server stub will convert the SOAP request into something the service implementation can understand (this is usually called *unmarshaling* or *deserializing*)
3. Once the SOAP request has been deserialized, the server stub invokes the service implementation, which then carries out the work it has been asked to do.
4. The result of the requested operation is handed to the server stub, which will turn it into a SOAP response.
5. The SOAP response is sent over a network using the HTTP protocol. The client stub receives the SOAP response and turns it into something the client application can understand.
6. Finally the application receives the result of the Web Service invocation and uses it.

#### 8.1.4.2.3 WSDL: Web Services Description Language

WSDL is a specification defining how to describe web services in a common XML grammar. WSDL describes four critical pieces of data: Interface information describing all publicly available functions. Data type information for all message requests and message responses. Binding information about the transport protocol to be used
Address information for locating the specified service
WSDL represents a contract between the service requestor and the service provider, in much the same way that a Java interface represents a contract between client code and the actual Java object. The crucial difference is that WSDL is platform- and language-independent and is used primarily (although not exclusively[11]) to describe SOAP services.
Using WSDL, a client can locate a web service and invoke any of its publicly available functions. With WSDL-aware tools, one can also automate this process, enabling applications to easily integrate new services with little or no manual code. WSDL therefore represents a cornerstone of the web service architecture, because it provides a common language for describing services and a platform for automatically integrating those services.
Some confusion sometimes occurs between web services and service oriented architecture. It should be noticed that describing a service by mean of WSDL is far from being an assurance that a system or application satisfies the SOA principle. Indeed, WSDL deals with the interface of a web service and has nothing to do with architecture. Many web services do not satisfy SOA principles, while SOA deals with architecture and do not impose anything on the technologies of implementation.

- *The WSDL Specification*

WSDL is an XML grammar for describing web services.
The specification itself is divided into six major elements:

[11] For instance, JBI uses WSDL.

### Definitions

The *definitions* element must be the root element of all WSDL documents. It defines the name of the web service, declares multiple namespaces used throughout the remainder of the document, and contains all the service elements described here.

### Types

The *types* element describes all the data types used between the client and server. WSDL is not tied exclusively to a specific typing system, but it uses the W3C XML Schema specification as its default choice. If the service uses only XML Schema built-in simple types, such as strings and integers, the *types* element is not required. A full discussion of the *types* element and XML Schema is deferred to the end of the chapter.

### Message

The *message* element describes a one-way message, whether it is a single message request or a single message response. It defines the name of the message and contains zero or more *message* part elements, which can refer to message parameters or message return values.

### portType

The *portType* element combines multiple message elements to form a complete one-way or round-trip operation. For example, a portType can combine one request and one response message into a single request/response operation, most commonly used in SOAP services. Note that a portType can (and frequently does) define multiple operations.

### Binding

The *binding* element describes the concrete specifics of how the service will be implemented on the wire. WSDL includes built-in extensions for defining SOAP services or REST services (in WSDL 2.0), and SOAP – or REST – specific information therefore goes here.

### Service

The *service* element defines the address for invoking the specified service. Most commonly, this includes a URL for invoking the SOAP service.
The next figure offers a concise representation of the WSDL specification.

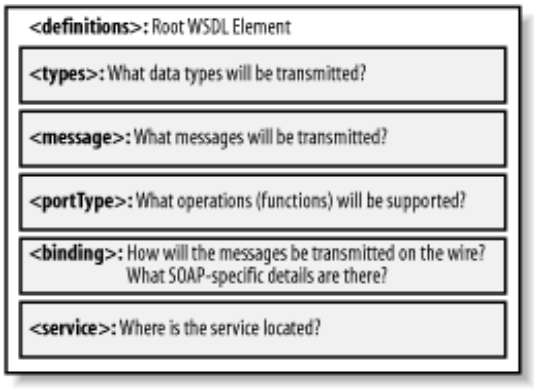


**Figure 42. The WSDL specification in a nutshell**

In addition to the six major elements, the WSDL specification also defines the following utility elements:

### Documentation

The *documentation* element is used to provide human-readable documentation and can be included inside any other WSDL element.

### Import

The *import* element is used to import other WSDL documents or XML Schemas. This enables more modular WSDL documents. For example, two WSDL documents can import the same basic elements and yet include their own service elements to make the same service available at two physical addresses. Note, however, that not all WSDL tools support the import functionality as of yet.

#### 8.1.4.2.4 WSDL 2.0

Web Services Description Language Version 2.0 (WSDL 2.0) provides a model and an XML format for describing Web services. WSDL 2.0 enables one to separate the description of the abstract functionality offered by a service from concrete details of a service description such as “how” and “where” that functionality is offered.

This specification defines a language for describing the abstract functionality of a service as well as a framework for describing the concrete details of a service description. It also defines the conformance criteria for documents in this language.

WSDL 2.0 describes a Web service in two fundamental stages: one abstract and one concrete. Within each stage, the description uses a number of constructs to promote reusability of the description and to separate independent design concerns.

At an abstract level, WSDL 2.0 describes a Web service in terms of the messages it sends and receives; messages are described independent of a specific wire format using a type system, typically XML Schema.
An *operation* associates a message exchange pattern with one or more messages. A *message exchange pattern* identifies the sequence and cardinality of messages sent and/or received, as well as whom they are logically sent to and/or received from[12]. An

[12] The objective here is not to replace a service registry, but to transmit some information (profile) about who is requesting the service, such information being for example likely to

*interface* groups together operations without any commitment to transport or wire format.
At a concrete level, a *binding* specifies transport and wire format details for one or more interfaces. An *endpoint* associates a network address with a binding. And finally, a *service* groups together endpoints that implement a common interface.
The figure below shows the conceptual WSDL component model.

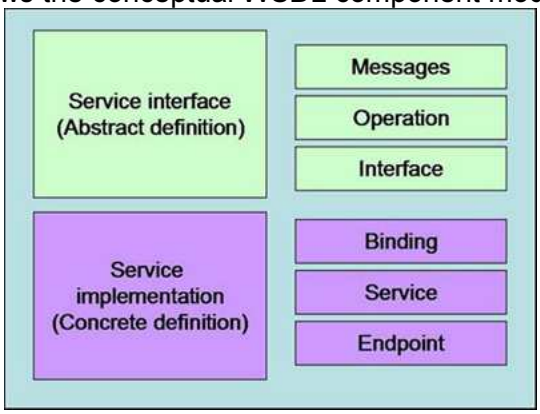


**Figure 43: WSDL Conceptual model**

8.1.4.2.5 WADL: Web Application Description Language

WADL (Web Application Description Language) is designed to provide a simple alternative to WSDL for use with XML/HTTP Web applications.
An increasing number of Web-based enterprises (Google, Yahoo, Amazon, Flickr to name but a few) are developing HTTP-based applications that provide programmatic access to their internal data. Typically these applications are described using textual documentation that is sometimes supplemented with more formal specifications such as XML schema for XML-based data formats. WADL is designed to provide a machine process-able description of such HTTP-based Web applications, especially those using XML.

Web service developers were stirred a few months ago in the technical media over the SOAP vs. REST debate, a now familiar theme which seems to come up every so often and one discussion which will surely never be completely settled given that each approach has its own technical merits on which to stand. But appropriate as each technique is for certain circumstances, until recently, there was one obvious part missing in RESTful approaches that was ever present in SOAP: the concept of a descriptor. Up next, we will explore an up and coming proposal named Web Application Description Language (WADL), which aims to provide descriptors for RESTful services.

For SOAP Web services, descriptors based on Web Services Description Language (WSDL) form a fundamental piece of their actual design, mainly on account of the underlying complexity present in the actual service. In these scenarios, a descriptor not only serves the purpose of formally describing all the business logic it can fulfill,

modulate the service provided (for example the quality of service). Getting some information of this kind is not in contradiction with the loose coupling concept of SOA.

but it also aids in the creation of helper classes – often called stubs – used to build service clients.

WADL is a similar description language to WSDL, but strictly targeting the requirements of RESTful services. REST started of simple enough, as live URL's on major portals which returned structured data to querying clients, but as interest has grown in this alternate approach to Web services, so has the scope and size of the business processes it attempts to fulfill, making descriptors a natural addition.

REST services being based on the back of HTTP can have the same methods – GET, POST, PUT, DELETE, and HEAD – available in this latter protocol. Input parameters on the other hand can also grow to be numerous for complex services, making the use of required, optional and type values a welcomed guide for developers trying to make sense of first time requests. When it comes to response values, structure responses have also grown in complexity, ranging from custom XML namespaces to JSON, no to mention the handling of fault scenarios in case a request is aborted.

#### *8.1.4.3 REST*

- *What is REST?*

REST stands for Representational State Transfer. It relies on a stateless, client-server, cacheable communications protocol. Actually, in virtually all cases, the HTTP protocol is used.

REST is an architecture style for designing networked applications. The idea is that, rather than using complex mechanisms such as CORBA, RPC or SOAP to connect between machines, simple HTTP is used to make calls between machines.

In many ways, the internet itself, based on HTTP, can be viewed as a REST-based architecture.

RESTful applications use HTTP requests to post data (create and/or update), read data (e.g., make queries), and delete data. Thus, REST uses HTTP for all four CRUD (Create/Read/Update/Delete) operations.
REST is a lightweight alternative to mechanisms like RPC (Remote Procedure Calls) and Web Services (SOAP, WSDL, et al.).
Despite being simple, REST is fully-featured; anything that Web Services can do can be done with a RESTful architecture.

REST is not a "standard". There will probably never be a W3C recommendation for REST, for example. And while there are REST programming frameworks, working with REST is so simple that you can often "roll your own" with standard library features in languages like Perl, Java, or C#.

- *REST as a lightweight Web services*

As a programming approach, REST is a lightweight alternative to Web Services and RPC.

Much like Web Services, a REST service is:

- Platform-independent (don't care if the server is Unix, the client is a Mac, or anything else),
- Language-independent (C# can talk to Java, etc.),
- Standards-based (runs on top of HTTP), and
- Can easily be used in the presence of firewalls.

Like Web Services, REST offers no built-in security features, encryption, session management, QoS guarantees, etc. But also as with Web Services, these can be added by building on top of HTTP:

- For security, username/password tokens are often used.
- For encryption, REST can be used on top of HTTPS (secure sockets).
- ... etc.

One thing that is *not* part of a good REST design is cookies: The "ST" in "RE**ST**" stands for "State Transfer", and indeed, in a good REST design operations are self-contained, and each request carries with it (transfers) all the information (state) that the server needs in order to complete it.

However, in recognition of the number of developers turning away from SOAP to the REST style, the W3C working group incorporated what they term "HTTP support" in the recently released WSDL 2.0[13] specification. This version, many years in the making, is a major revision of version 1.0 with reorganization supposed to reduce complexity. In theory, a WSDL 2.0 document can describe a service as both a SOAP and a REST application. The tools previously used to create Web services from WSDL documents will need major modification to work with WSDL 2.0.

As a conclusion, for a programmer when it is applicable, REST is a simpler way to code web services than SOAP.

#### *8.1.4.4 SCA: Service Component Architecture*

##### 8.1.4.4.1 Overview

Service Component Architecture (SCA) is a set of specifications which describes a model for building applications and systems using a SOA. SCA extends and complements prior approaches to implementing services, and thus, simplifies application development and implementation developed using SOA.
SCA encourages an SOA organization of business application code based on components that implement business logic, which offer their capabilities through service-oriented interfaces called services and which consume functions offered by other components through service-oriented interfaces, called references. SCA divides up the steps in building a service-oriented application into two major parts:

- The implementation of components which provide services and consume other services
- The assembly of sets of components to build business applications, through the wiring of references to services.

[13] See : http://www.w3.org/TR/wsdl20/

SCA emphasizes the decoupling of service implementation and of service assembly from the details of infrastructure capabilities and from the details of the access methods used to invoke services. SCA components operate at a business level and use a minimum of middleware APIs.
SCA supports service ***implementations*** written using any one of many programming languages, both including conventional object-oriented and procedural languages such as Java™, PHP, C++, COBOL; XML-centric languages such as BPEL and XSLT; also declarative languages such as SQL and XQuery. SCA also supports a range of programming styles, including asynchronous and message-oriented styles, in addition to the synchronous call-and-return style.
SCA supports ***bindings*** to a wide range of access mechanisms used to invoke services. These include Web services, Messaging systems and CORBA IIOP.
Bindings are handled declaratively and are independent of the implementation code.
Infrastructure capabilities, such as Security, Transactions and the use of Reliable Messaging are also handled declaratively and are separated from the implementation code. SCA defines the usage of infrastructure capabilities through the use of ***Policies*** and ***Profiles***, which are designed to simplify the mechanism by which the capabilities are applied to business systems.
SCA also promotes the use of Service Data Objects (SDO) to represent the business data that forms the parameters and return values of services, providing uniform access to business data to complement the uniform access to business services offered by SCA itself.
The SCA specification is divided into a number of documents, each of them dealing with a different aspect of SCA.

- The ***Assembly Model*** deals with the aggregation of ***components*** and the linking of components through wiring using ***composites***. The Assembly Model is independent of implementation language.
- The ***Client and Implementation*** specifications deal with the implementation of services and of service clients – each implementation language has its own Client and Implementation specification, which describes the SCA model for that language.

#### 8.1.4.4.2 Advantages of SCA

SCA simplifies the creation and integration of business applications built using a SOA.
SCA provides a mechanism to build coarse-grained components as assemblies of fine-grained components.
SCA relieves programmers from the complexity of traditional middleware programming by abstracting it from business logic. It allows developers to focus on writing business logic and can free them from the need to spend significant cycles on more low-level implementation techniques.
Some advantages of the SCA approach are:

- Simplified business component development
- Simplified assembly and deployment of business solutions built as networks of services
- Increased portability, reusability, and flexibility
- Protection of business logic assets by shielding from low-level technology change

- Improved testability.

SCA offers a mechanism to package and deploy sets of closely related components, which are developed and deployed together as a unit. It decouples service implementation and assembly from the details of infrastructure capabilities and from the mechanisms for invoking external systems. This enables portability of services between different infrastructures.

#### 8.1.4.4.3 Service implementations and service clients

Service implementations are concrete implementations of business logic, which provide services and/or consume services. The implementation is the servant of the business process.
An implementation can provide a service, which is a set of operations defined by an interface used by other components. Implementations can also use other services, called service references, which indicate the implementation's dependency on services provided elsewhere. An implementation may also have one or more configurable properties. A property is a data value that can be externally configured and affects the business function of the implementation.
SCA services typically use document-style business data for parameters and return values, and preferably these parameters are represented using as Service Data Objects (SDOs) (please refer to the Resources section for more information).
Services, references and properties are the configurable aspects of an implementation – SCA refers to them collectively as *Component type*.
Configuring a reference is done by binding the reference to a target service, which will then be used by the implementation when it invokes the reference. Configuration of a property involves setting a specific data value for the property. In a SCA framework, one implementation can be used to build multiple different components, with each component having a different configuration of the references and properties.
Components and their services are used by other local components, or components can be used for remote access.

#### 8.1.4.4.4 Assembly

Assembly is the process of composing business applications by configuring and connecting components that provide service implementations. SCA assembly operates at two levels:

- Assembly of loosely connected components within a system
- Assembly of loosely connected components within a module

The SCA assembly model consists of a series of artifacts, which are defined by XML elements.

#### 8.1.4.4.5 Module assembly

A *SCA module* is the largest composition of tightly-coupled components that are developed and deployed together into a *SCA system*. It is the basic unit of loosely-coupled composition within a SCA System. A SCA module contains a set of

components, external services, entry points, and the wires that interconnect them. Modules contribute to the implementation of services in an SCA System.
*Entry points* define the public services provided by the module, which can either be used by other components within the same module or which can be made available for use outside the module. These are used to publish services provided by a module using a specified *binding*.
*External services* within a module represent remote services provided by other modules. They are external to the SCA module that uses the service. These external services can be accessed by components within the module like any service provided by a SCA component. External services use bindings to describe the access to external services.
The interface of an external service must be remotely usable.
This is illustrated in Figure 44. In this figure, the term "composite" is equivalent to the term "module" of the 0.9 version of the specifications.

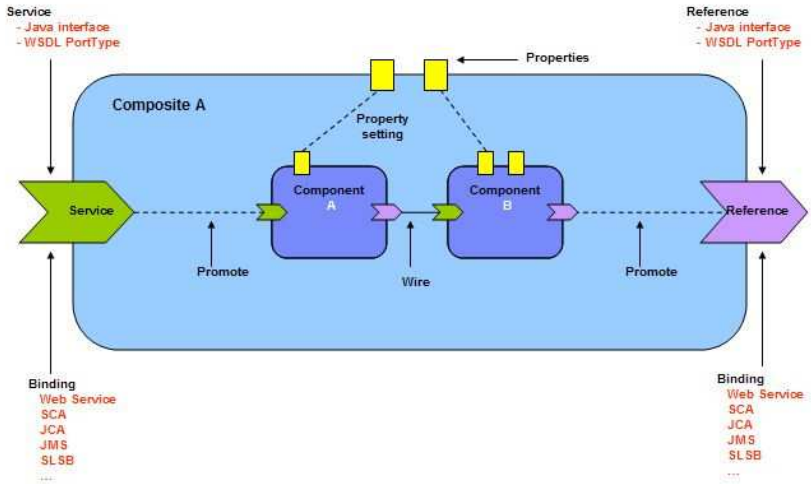


**Figure 44: SCA assembly**

### 8.1.4.4.6 System assembly

A *SCA subsystem* is used to group modules which provide related business functions through the configuration and administration of module components, external services and entry points, plus the wires that interconnect them in a *SCA system*. The configuration of a SCA system is represented by the combination of all the subsystems deployed into it. Figure 45 below is an example of system assembly; it illustrates how sub-systems and modules are wired using services and references.

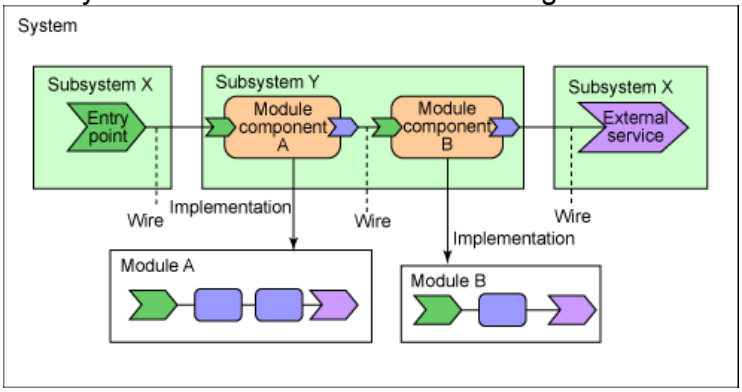


**Figure 45: Service Component Architecture for system and module assembly**

Let's break down what can be seen in the previous figure

- **Module components** in a subsystem configuration represent configured instances of a SCA module, where the module component can set values for the external services of the module and can set values for properties exposed by the module.
- **External services** are remote services that are external to the SCA system that uses the service. The differences between an external service on the module level and on the system level are:
  - The name has to be unique across all external services defined in the subsystem
  - No external service in a subsystem can have the same name as a module component since they both can be targets of wires
- **Entry points** are used to declare the externally accessible services of a subsystem. They are used by other organizations or customers as a Web service. The differences between entry points on the module level and on the system level are:
  - The name has to be unique across all entry points defined in the subsystem. No entry point in the subsystem can have the same name as a module component
  - The specification of the reference child element is optional, since it can be wired through a wire supplied by another subsystem

#### *8.1.4.5 Using dataflow model for image processing*

In the context of multi-core processors and the trend toward many-core, dataflow programming can be used as a solution to the parallelization problem. By decoupling computation from communication, this paradigm naturally exposes parallelism in several ways. Moreover, a large number of applications are more easily expressed in this programming model into an imperative concurrent model (i.e C + threads) because they are naturally data oriented. Images encoding and decoding, images processing, data encryption and decryption, software defined radio are good example of dataflow applications. In this programming model, applications are described as a graph of actors connected by the mean of fifo (first in first out) channels. This graph of actors takes as input a stream of data to produce a new stream of data as output. The input stream may be potentially infinite. Actors operate computation on the data coming on their input channels to produce new data on their output channels.

According to the dataflow semantic, an actor can be fired, each time enough data are present on its input channels. Channels are the only way allowing actors to communicate. As a consequence, actors can be freely executed in parallel once required data on input channels are available. See ([Lee 95] . A dataflow compiler is responsible to translate a dataflow program to binary code for a given architecture.

Many compilers first translate the dataflow program into an imperative concurrent program (i.e C + threads) and then rely on standard compilation tools (i.e Gcc) to produce the final binary. Depending on the targeted architecture, dataflow compilers can exploit many programs transformations allowing generating code with the right degree of parallelism. For example, if there is not enough parallelism expressed in the

initial application, the compiler can parallelize stateless actors (actors without any states between two firings) to allow them to work on different sets of data simultaneously. The compiler can also exploit so called pipeline parallelism resulting from implicit loop over the actors' graph. The compiler can generate code that will let earliest actors in graph to work on new data even if later actors have not finished their work for previous data.

Many specializations of this model for different applicative contexts have been proposed. The main goal of these specializations is to provide additional information to the compiler allowing producing more efficient code while still providing enough expressiveness for the considered application class.

A widely used specialization is the Synchronous Dataflow (SDF) one [Lee 87] . In this model, the number of input data consumed and produced each time an actor is fired must be provided to the compiler and must remain constant during all the application's execution. This information allows the compiler to statically compute actors firing sequences and to statically compute the maximum size required for every fifo channel in the application.

The compiler can also detect unbounded fifo channels requirements and applications that will deadlock because of a lack of data production.

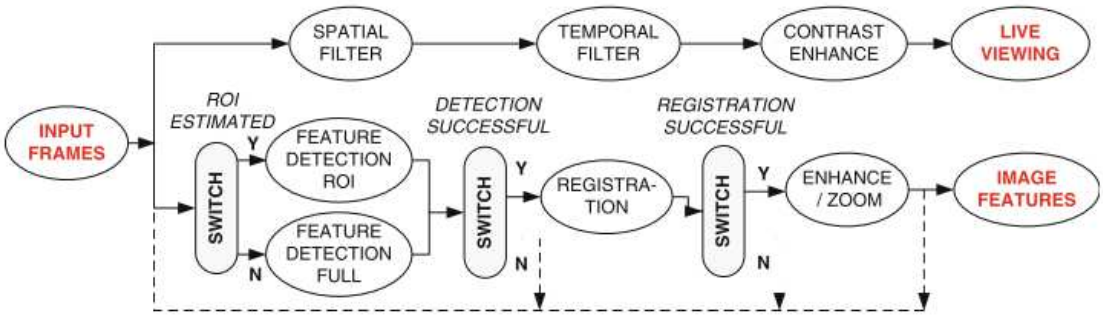


**Figure 46: An example of dataflow application for medical imaging**

Figure 46 shows an example of a dataflow application in the context of medical imaging. This is the dataflow graph of an interventional X-ray application to detect and enhance moving features, combined with several image quality functions.

Each node in the graph is an actor performing some computation on its input data. This graph belongs to the general dataflow model because actors' consumption and production rates are not specified. It also contains some special actors called switch, allowing to conditionally rooting data to different actors.

## 8.2 References

[Era 07] Eral, T. (July 2007). SOA, Principle s of Service Design.
[Int 09] Intel, I. (2009). Architecting Software as a Service for Entreprise.
[Lee 87] Lee, E., & Parks, T. (1987). Synchronous data flow. Proceedings of the IEEE.
[Lee 95] Lee, E., & Parks, T. (1995). Dataflow process networks. Poceedings of the IEEE, vol 75.

[MAR 12] Iain James Marshall, L01.3 - AccordsPlatform v1.2.pdf, 2012, CompatibleOne project.
[Oas 06] OASIS, O. C. (2006). Reference Model for Service Oriented architecture 1.0.
[Sch 08] SCHMELZER, R. (Feb. 28, 2008). Why Service Consumers and Service Providers Should Never Directly Communicate. ZAPFLASH-2008228.

# 9 Abbreviations and Definitions

| Term | Meaning |
|---|---|
| $^{18}$F-FDG | $^{18}$F-fluoro-deoxyglucose |
| Adaptive | Tumor segmentation method based on PET modality [VAU 09] |
| API | Application Programming Interfaces |
| $B_{avg}$ | Averaged background measured activity |
| Black | Tumor segmentation method based on PET modality [BLA 04] |
| CPU | Central Processing Units |
| CT | Computed Tomography |
| DSPs | Digital Signal Processors |
| Fitting | Tumor segmentation method based on PET modality [TYL 10] |
| FPGA | Field Programmable Gate Array |
| FWHM | Full Width at Half Maximum |
| GPU | Graphics Processing Units |
| GTV | Gross Tumor Volume |
| Nestle | Tumor segmentation method based on PET modality [NES 05] |
| Max | Tumor segmentation method based on PET modality *(*[ERD 97]*,*[PAU 04]*,* and [NES 07]*)* |
| $Mean_{int}$ | Mean intensity |
| MPI | Message Passing Interface |
| NM | Nuclear Medicine |
| NSCLC | Non-Small Cell Lung Cancer |
| OpenCL | Open Computing Language. |
| OpenGL | Open Graphics Library |
| PET | Positron Emission Tomography |
| PVE | Partial Volume Effect |
| ROI | Region of Interest |
| $Th_{opt}$ | Optimal threshold |
| TPS | Treatment Planning System |
| SIMD | Single Instruction Multiple Data |
| SUV | Standard Uptake Value |
| Variational | Tumor segmentation method based on both CT and PET modalities [WOJ 10] |
| VOI | Volume Of Interest |
| VCL | Virtual OpenCL Layer |

| Term | Meaning |
|---|---|
| WebCL | Web Computing Library |
| WebGL | Web Graphics Library |

## 9.1 Response Evaluation Criteria in Solid Tumors (RECIST)

A published guideline [THE 00] used to assess the change in size (maximum diameter in 2D plane) of solid tumors and lymph nodes in response to therapy.
The response is categorized into:
• Complete Response (CR): disappearance of all target lesions
• Partial Response (PR): 30% decrease in sum of diameters of all target lesions
• Stable Disease (SD): No change in the size of the lesions
• Progressive Disease (PD): 20% or absolute 5 mm increase in sum of diameters of target lesions

The user identifies the target lesions on the baseline examination. Target lesions should be selected on the basis of their size (lesions with the longest diameter) and their suitability for accurate repeated measurements (either by imaging techniques or clinically).

## 9.2 World Health Organization (WHO)

The WHO criteria defined shrinkage of a tumor as the decrease in the product of the largest perpendicular diameters in the largest “slice” of the tumor on a scan.