CompAS: a New Approach to Commonality and Variability Analysis with Applications in Computer Assisted Orthopaedic Surgery

Gisèle Douta [1], Haydar Talib[1], Oscar Nierstrasz[2], Frank Langlotz[1]

[1] MEM Research Center for Orthopaedic Surgery, University of Bern, Switzerland
[2]Software Composition Group, University of Bern, Switzerland

Keywords: computer-assisted surgery, software reuse, component-based programming, domain analysis, commonality and variability, software evolution

Preprint. Please cite as:

Gisèle Douta, Haydar Talib, Oscar Nierstrasz and Frank Langlotz, "CompAS: A new approach to commonality and variability analysis with applications in computer assisted orthopaedic surgery," Information and Software Technology, vol. 51, 2008, pp. 448-459. DOI: 10.1016/j.infsof.2008.05.017

MEM Research Center for Orthopaedic Surgery
Institute for Surgical Technology & Biomechanics
University of Bern
Stauffacherstrasse 78
CH-3014 Bern Switzerland
Tel: +41 31 631 5959
Fax: +41 31 631 5960

Software Composition Group
Institute of Computer Science
University of Bern
Neubrückstrasse 10
CH-3012 Bern Switzerland
Tel: +41 31 631 4692
Fax: +41 31 631 3355

Abstract:
In rapidly evolving domains such as Computer Assisted Orthopaedic Surgery (CAOS) emphasis is often put first on innovation and new functionality, rather than in developing the common infrastructure needed to support integration and reuse of these innovations. In fact, developing such an infrastructure is often considered to be a high-risk venture given the volatility of such a domain. We present CompAS, a method that exploits the very evolution of innovations in the domain to carry out the necessary quantitative and qualitative commonality and variability analysis, especially in the case of scarce system documentation. We show how our technique applies to the CAOS domain by using conference proceedings as a key source of information about the evolution of features in CAOS systems over a period of several years. We detect and classify evolution patterns to determine functional commonality and variability. We also identify non-functional requirements to help capture domain variability. We have validated our approach by evaluating the degree to which representative test systems can be covered by the common and variable features produced by our analysis.

1 Introduction

Computer-Assisted Orthopaedic Surgery (CAOS) is a technological domain that arose in the early 1990s from the combination of several other mature sciences such as image processing, biomechanics, computer graphics, and robotics. Orthopaedic surgical procedures follow the common basic principle of "placing an object (guide wire, screw, tube or scope) at a specific site, via a trajectory which is planned from medical images and governed by three-dimensional anatomical constraints" [1]. In order to provide surgeons with a means to perform these procedures with higher accuracy, CAOS systems have been progressively introduced into the operating room. Using virtual representations of the surgical instruments and of the operated anatomy, CAOS systems replay in real time the surgeon's actions on a computer screen (Fig. 1). Although many technical approaches have been taken to develop these systems their conceptual designs remain similar: CAOS systems typically consist of a planning subsystem to help the surgeon define the optimal surgical strategy and a navigation subsystem to support him or her in achieving the planned strategy [2, 3].
Because of the commonality in surgical gestures the variety of CAOS systems developed to assist in diverse orthopaedic surgeries offer common features such as loading/acquisition of medical data, data visualization in 2D and/or 3D, and selection of the best fitting implant. However, up to now each application is considered as an individual system strictly bound to a specific surgical procedure and pathology. Such a system engineering approach results in monolithic systems that do not have the flexibility required to allow one to take advantage of the functional similarities of these systems through software reuse.

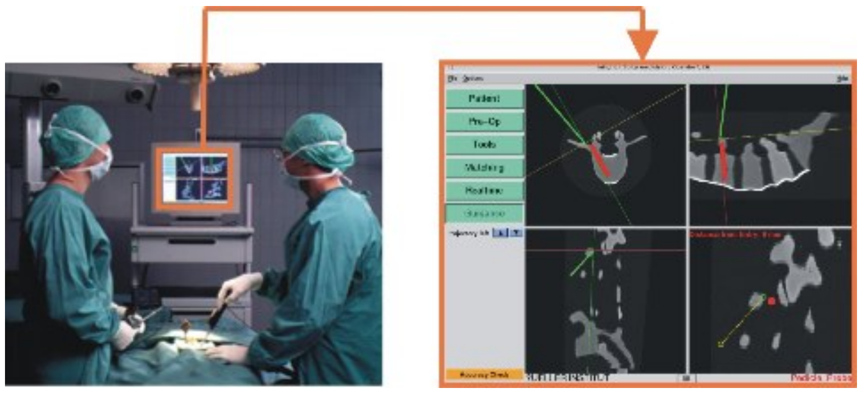


Fig. 1 Computer-Assisted Orthopaedic Navigation

The basic idea underlying software reuse is simple: rather than building software systems from scratch we assemble them from common reusable assets such as modules, objects and classes. Component-based programming is a recently-established paradigm for software reuse. According to Szyperski [4] "a software component is a unit of composition with contractually specified interfaces and explicit context dependencies only. A software component can be deployed independently and is subject to composition by third parties'". In other words components are the building blocks from which an application can be composed in a "plug and play" manner. Adopting such a software development approach implies a move from single systems engineering to families of systems. A system family is a set of software applications sharing a large number of common properties [5]. Domain engineering refers to methods for defining, designing and implementing the necessary assets to support software reuse in system families. The initial and crucial step of these software engineering methodologies is called domain analysis. It aims at identifying commonalities, variabilities and dependencies in the selected family and at integrating them in a coherent model [6].

In order to take advantage of the functional similarities present in the CAOS family of applications, we propose to apply component-based programming to the development of CAOS systems. Because it is the necessary prerequisite to enable efficient component-based software reuse we focused first on domain analysis. We have designed CompAS, a new approach to commonality and variability analysis to support component-based architectural modeling. The key novelty of our approach is to analyze the evolution of the domain to effectively determine which features should be included as common or variable.

2. Challenges in Performing Domain Analysis in CAOS

A domain model is the set of artifacts resulting from the domain analysis. The appropriate domain model is the one that provides the most sensible system decomposition in terms of common and variation points. Its achievement requires a careful balance between current and future needs. This information can usually be extracted from interviews with domain experts, existing systems, and literature. Yet the software development context considered here is a research environment where, contrarily to the usual industrial approach, the most common practice is to implement prototype applications more or less from scratch, in order to allow the clinical validation of the investigated concepts, which usually implies inconsistent system implementation documentation. Moreover, among the potential candidates for the investigated family of applications only a restricted number of them were implemented at our institute. This means that we had access to the code of only few of our application family's members. However, CAOS has the particularity to be a domain for which research and industry

are still not only continuously innovating but as well publishing these innovations. We propose a method that takes advantage of this extended and publicly available literature to palliate our lack of systems documentation.
The identification of commonalities and variabilities mainly relies on the capabilities of the domain analyst to abstract from and refine the collected data and knowledge. We propose to strengthen the process of commonalities and variabilities identification with a quantitative evaluation of functional evolutionary trends. Several methodologies have been proposed to evaluate software evolution, one of the main differences between them being the type of data they require as input. Some methods extract evolution trends from version control data such as that provided by the Concurrent Version System CVS [7, 8]. This information (e.g. modification reports), can be combined with problem reports extracted from a bug tracking system and with feature information derived from the executable itself to visualize feature evolution [9]. In our case where only scarce source code data are available, we were inspired by the telephony feature evolution study performed by Anton et al based on publicly available information about telephony [10] to consider literature as our data source. We suggest using evolution matrices, which track the evolution of features over time, to expose implicit patterns in natural lifecycle of features [11].
Software family engineering not only focuses on currently existing systems but it anticipates future needs and variations as well. The results of the domain analysis must then appropriately model variations so that it provides:

- the software user with an explicit and concise representation of available variabilities.
- the developer of reusable software with the knowledge why a certain variation point is included in the software.
- the software architect with the basis to design an architecture flexible enough to support the family diversification and evolution.

Apart from the desire of continuously proposing more appropriate and useful functionalities, CAOS research also aims at providing innovative methods and technology to implement these functionalities. In order to model CAOS variability at the functional and technological level we propose a taxonomy of change scenarios. By taxonomy we simply mean the dictionary definition of "a system for naming and organizing things ... into groups, which share similar qualities" [12]. By change scenarios we refer to situations, where only a particular functional or technological aspect of an existing system is modified.

3. Domain Analysis

Domain analysis is the step of domain engineering during which the domain analyst selects a family of applications (or domain) to study, collects the domain knowledge, organizes it into a set or artifacts (domain models) describing the common and variable properties of the system family, and defines the semantics of these properties and the dependencies between them. A large number of domain analysis methods exist and all of them agree that the appropriate source of information should mainly come from [6, 13, 14]:

- human sources: domain experts, system users, developers, etc.
- existing systems: source code, design documentation, user manuals, etc.
- literature: books, articles, standards, etc.

Assimilating information coming from such diverse sources to create a coherent domain model is a difficult task. This is why the choice of the sources of data that will most efficiently lead to the appropriate domain model is nevertheless left to the domain analyst's discretion. The foundation of software reuse is the discovery and exploitation of commonality across related software systems; commonality and variability identification is the central component of a domain analysis, which is therefore a key to successful software reuse.

Most of the time the proposed commonality and variability differentiation method consists of identifying aggregation/decomposition and generalization/specialization relationships among the identified reusable assets [15-21]. These relationships are also referred to as "is a", "consist of", "kind of" or "whole-part" relationships. This approach results in a set of hierarchical diagrams, the most popular one being the feature diagram (Fig. 2) proposed by the FODA method [15]. One other alternative is to use lexical analysis. ODM [16], for example, suggests that the domain analyst should identify terms that play the same semantic role in the domain, and that he or she should define semantic relationships among these terms so that features correspond to sentences or statements in this defined language. As for DARE [22], it offers a tool suite that includes support to automatically extract and cluster words according to their conceptual similarity using the provided textual domain data.

Since the previously mentioned diagrams also implicitly model variability, not all methods have a specific means to capture variability. However, the domain analysis component of PuLSE [18] mentions the design of a decision hierarchy where each type of variation in the domain will match a decision type. For FODA [15] and RAPID [21], variability can be characterized using templates and parameterization to capture variation context. Finally, Gomaa [20] in his approach to domain analysis uses change scenario impact analysis, where one can trace the necessary variants to support a defined change scenario.

Certain domain analysis methods propose an algebraic approach [23, 24, 25] where the main idea is to formalize domain knowledge in the form of a network of related algebraic specifications. However, these methods do not contain an explicit commonality and variability identification phase.

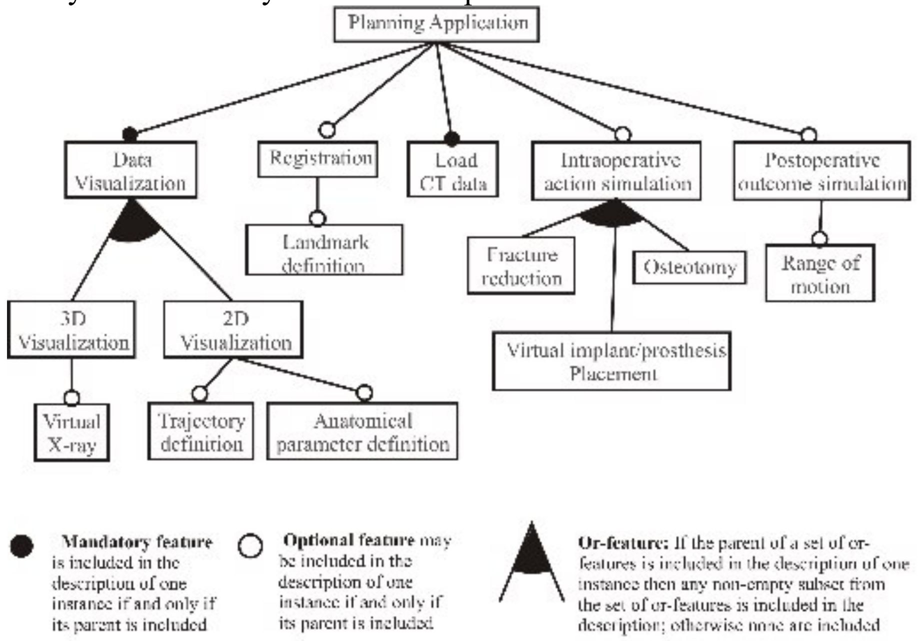

Planning Application
Data Visualization
Registration
Load CT data
Intraoperative action simulation
Postoperative outcome simulation
Landmark definition
Fracture reduction
Osteotomy
Range of motion
3D Visualization
2D Visualization
Virtual implant/prosthesis Placement
Virtual X-ray
Trajectory definition
Anatomical parameter definition
Mandatory feature is included in the description of one instance if and only if its parent is included
Optional feature may be included in the description of one instance if and only if its parent is included
Or-feature: If the parent of a set of or-features is included in the description of one instance then any non-empty subset from the set of or-features is included in the description; otherwise none are included

Fig. 2 Feature Diagram for Computed Tomography Based Planning Applications

4. The CompAS approach

In certain domains such as CAOS, data coming from existing systems are not consistent enough to be exploited for domain analysis. Moreover, as already noted for other domains [14], CAOS can also be seen as a business area. "Such a domain not only contains applications, it is constrained by external forces that motivate the domain". We therefore designed an approach relying on an intensive literature review and regular domain expert interviews and exploiting the business area characteristic of a domain to capture variation. What we term CompAS (Commonality and Variability Analysis to Support Component Based Architectural Modeling) is an approach based on the two following hypotheses:

- There is a correlation between the functional system evolution and functional commonality and variability properties.
- In order to understand and capture variation, it is also necessary for the non-functional requirements that constrain the domain to be identified.

Consequently, the presented method is divided into two phases. First, we compute evolution matrices to identify functional evolutionary trends and exploit evolution patterns to differentiate common and variable features. Second, we identify the domain's non-functional requirements and use them to support the capture of the domain variability. The two parts of the method are independent; the domain analyst is free to choose to apply either or both of them. The remaining of this section gives an overview of CompAS method that we illustrate later in section 5 with the CAOS case study.

4.1 The data source

The data source for CompAS is an extensive set of descriptions that is representative of the various types of system implementations and/or evolutions for the considered application family during the evaluated period of time. The data source should contain functional system descriptions to support the first part of the approach and/or textual justification of specific system evolutions to support the second part. The system descriptions have to be detailed enough to allow one to perform correct, consistent and complete functional decomposition. In order to ensure these qualities CompAS suggests to the domain analyst to regularly consult domain experts. The role of experts here should be to ensure a pertinent feature selection and relevant non-functional constraint identification for the domain. They should as well provide the domain analyst with sufficient domain knowledge to allow him or her to deduce the presence or not of features in a system based on the most likely types of system descriptions.

4.2 Evolution matrix

The concept of software evolution matrices has been introduced by Lanza in 2001. We briefly describe here the conceptual principle, and for further details the readers should refer to the related publications [11, 26]. An evolution matrix combines software visualization and software metrics in order to reduce the complexity of the comprehension of a large amount of data and to provide quantitative evaluation of software evolution. It is an organized disposition of rectangular shapes, where each rectangle is the visualization of up to five metrics: the rectangle's position represents two metrics, the width and height can encode two others metrics, and finally the rectangle's color can be used to render a fifth metric. For CompAS each element of the matrix depicts the computed metrics for a given year (on the abscissa) and a defined functionality (on the ordinate). The width reflects the cumulative percentage of systems including the evaluated feature while the height represents the distribution over years of systems containing this feature. We use a variation of gray to represent the percentage of system descriptions that belong to the considered family of applications for a given year (Fig. 3). When the analyzed family can be divided in sub-families, which are subsets of instances that share more functional characteristics than with the remaining of the family, one matrix per sub-family should be designed.

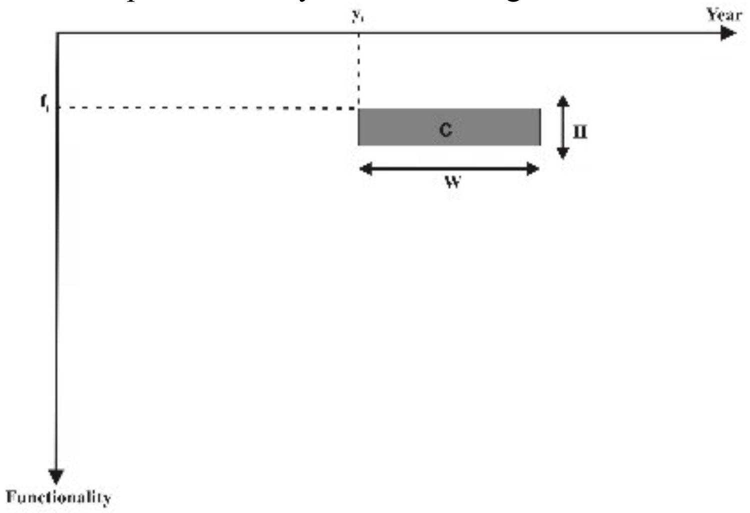


Fig. 3 Evolution matrix principle

If we call M the number of sub-families, I the length in years of the evaluation period, β_{im} the number of systems that belong to the sub-family m in the year y_i, and α_{ij}^{m} the number of systems in the sub-family that contain the feature f_j in the year y_i then:

The rectangle width W is proportional to $w_{ij} = \frac{\sum_{k=0}^{i} \alpha_{kj}^{m}}{\sum_{k=0}^{i} \beta_{km}}$ *which evaluates the proportion of systems in the sub-family m that contain the feature j up to the year i.*

The rectangle height H is proportional to $h_{ij} = \frac{\alpha_{ij}^{m}}{\sum_{k=0}^{I} \alpha_{kj}^{m}}$ *which evaluates among all the systems having the feature j in the sub-family m, the proportion that belong to the year i.*

The rectangle color C is relative to $c_{ij} = \frac{\beta_{im}}{\sum_{k=0}^{M} \beta_{ik}}$ *which evaluates the proportion of systems that belong the sub-family m in the year i, relative to the other sub-families*

Evolution matrices are defined along with a terminology to characterize evolution. For the terminology to be usable in commonality and variability analysis we have adapted it to functional evolution as presented in Table 1.

Evolution pattern	***Common and variable properties***
Presence-based pattern : defined by the evolution of the rectangle height	
Dayfly: Feature that has a very short lifetime (one or two consecutive years).	A variation that did not yet make a breakthrough or that has been abandoned.

Evolution pattern	***Common and variable properties***
Persistent across sub-families: Feature that is present during the entire evaluation period and in all sub-families.	Common core of the domain
Persistent in a given sub-family: Feature that is present during the entire evaluation period of a given sub-family.	Common feature of the sub-family
Shape-based pattern: defined by the evolution of the rectangle width	
Red Giant: A feature that keeps on being very wide over time.	A common feature
White Dwarf: A feature that used to be of a certain width but slowly decreases.	A common feature, which decreases in popularity to become variable.
Supernova: a feature that suddenly explodes in width, and eventually becomes a Red Giant.	A variation, which grows in popularity and eventually becomes a common feature.
Idle: A feature that remains relatively small over time.	A variation rarely used or specific to a certain type of application in the sub-family.
Pulsar: A feature that grows and shrinks repeatedly during its lifetime.	A variation with "unstable" popularity

Table 1: Correlation between Evolution Patterns and Commonality and Variability

4.3 Capturing domain variation by identifying "evolution factors"

In order to model variation CompAS proposes using a taxonomy whose definition is based on the data part that contains descriptions and justifications of system evolutions. During this phase, the domain analyst needs to identify what we call "evolution factors", namely the non-functional requirements that constrain the domain and therefore drive innovation. The taxonomy resulting from CompAS has two levels of categorization. The first level of categorization is the list of features that the gathered data propose to modify and improve. To define the second level of categorization the domain analyst needs to use his or her personal domain knowledge and the one he or she can obtain from domain experts, to identify the non-functional requirements that regularly motivate the implementation of new functional variations. For each first level category he or she should then estimate which of the identified evolution factors are the more influential. CompAS suggests here to evaluate the percentage of the data whose contribution influences the domain evolution with regard to a given evolution factor. Finally the domain analyst should divide each first level category into subcategories that correspond to the type of functional or technological changes that result from a wish to fulfill the identified most influential evolution factors.

4.4 The CompAS process

Applying CompAS consists then in the following activities.

A. Functional evolution based commonality and variability identification:
    1. Gather a set of system functional descriptions
    2. Extract from the data a set of features, i.e. a set of user-visible system functionalities
    3. Compute and build the family (or sub-families) evolution matrix(es)
    4. Identify the evolution pattern followed by each feature and deduce from it their commonality and variability property

B. Business-oriented variation capture:
  1. Gather a set of functional evolution descriptions
  2. List the features concerned by the gathered descriptions
  3. For each feature define the most influential evolution factors and the resulting functional and technological variation
  4. Use the results of 2 and 3 to build the taxonomy

5. Applying CompAS to CAOS

5.1 Data source

The annual meeting of the International Society for Computer Assisted Orthopaedic Surgery is the main conference in our domain of interest. This meeting is a practice- and clinically-oriented conference where technologists and surgeons gather to exchange information of an investigative and clinical nature. This conference started in 2001 as the result of the fusion of two former events: The CAOS symposia held in Switzerland from 1995 to 2000 and CAOS/USA, the North American program held annually from 1997 to 2001. We thoroughly reviewed proceedings of this conference in order to extract useful information for our domain analysis. However, because the CAOS Symposia only published short abstracts that did not contain enough details to support our analysis, we excluded these proceedings. Consequently, we had available to us a total of 1076 abstracts covering the years 1998-2005 (CAOS/USA proceedings for 1997 was missing in our collection).

5.2 Commonality and variability identification

5.2.1 System functional descriptions

In our collection of proceedings we selected all the abstracts containing a functional description of a system and decomposed each of them into a set of features. CAOS systems are usually described by listing the different assisting elements they provide. Such a characteristic implies easy feature decomposition where a feature is a set of functionalities provided to assist the user in performing one step of the computer-assisted surgical procedure (trajectory definition, image segmentation, etc.). CAOS system users (i.e. surgeons) and CAOS systems developers within and outside our institute were consulted to evaluate the pertinence of our set of features. The collected system descriptions have been divided into four sub-families. Three of them are defined by the anatomical area operated by the assisted surgery (spine, hip-pelvis, and knee). The fourth one is defined by the type of surgery assisted (traumatology).
We identified 137 system descriptions with a major percentage of them (40%) belonging to the knee family. Their distribution over our four identified families is presented in Fig. 4.

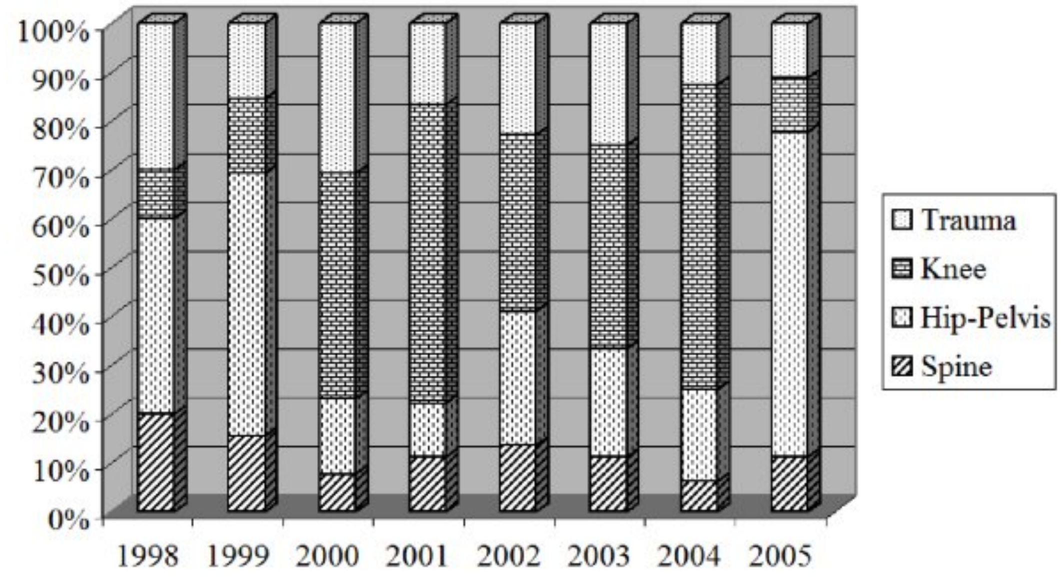


Fig. 4 System Description Distribution

5.2.2 Features and evolutions matrices

For the spine, hip-pelvis, knee and traumatology sub-families we identified respectively a set of 13, 16, 21 and 14 features, the identified feature list for the hip-pelvis and knee family can be visualize on the ordinate of the matrices presented in Fig 5 and Fig 6 while the complete feature set appears in table 2. We computed the necessary metrics and implemented, using Qt (Trolltech AS, Oslo, Norway) [27] a prototype support tool to visualize the resulting evolution matrices. This tool takes as input the computed metrics, and displays the resulting matrices as shown in Fig 5. and Fig 6. for the hip-pelvis and knee families

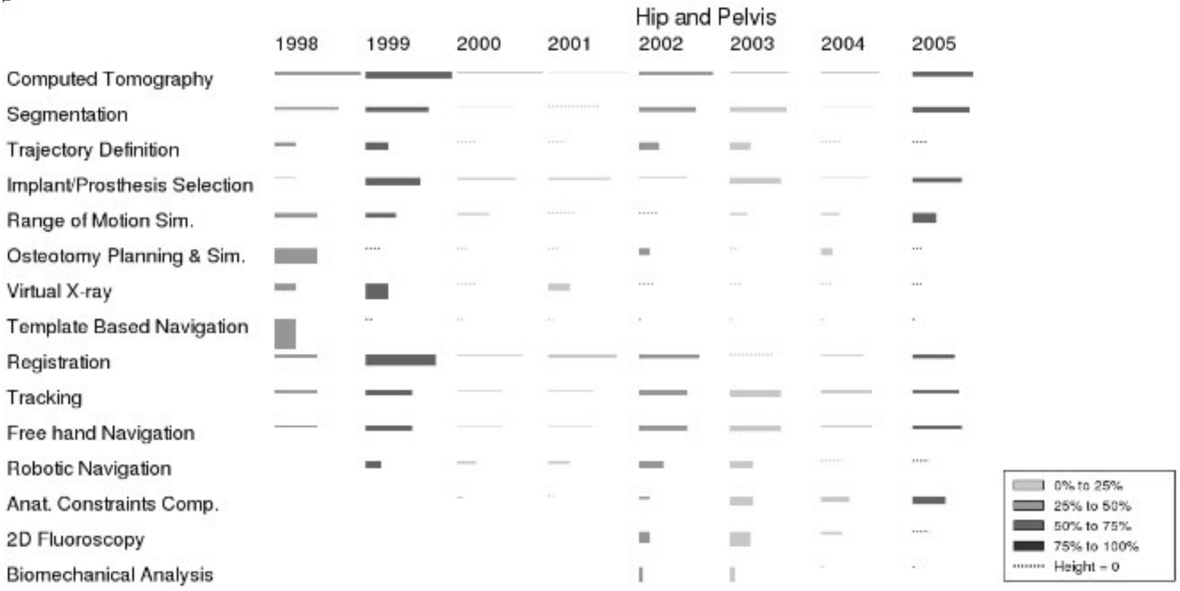


Fig. 5 Evolution Matrix of the Hip-Pelvis Family

The previously given metrics definitions imply that for a given column all the rectangles have the same color. As a consequence this allows a straightforward visualization of how prevalent each family has been for CAOS research over time. We can see that 1999/2005 and 2001/2004 were years when research was more involved in the development of hip-pelvis and knee applications respectively. Deducing the year in which a new technology has been introduced is as well obvious since functionalities are gradually added at the bottom of the matrix. We can therefore state that the introduction of 2D fluoroscopy, which was a major contribution to CAOS development, took place between 2001 and 2002 for both families. We can as well notice that the knee family has been in constant evolution from 1998 to 2003: each year, new functionalities have been introduced. While the hip-pelvis family had a more moderate evolution, indeed only a very small number of features had been introduced and in a shorter period of time.

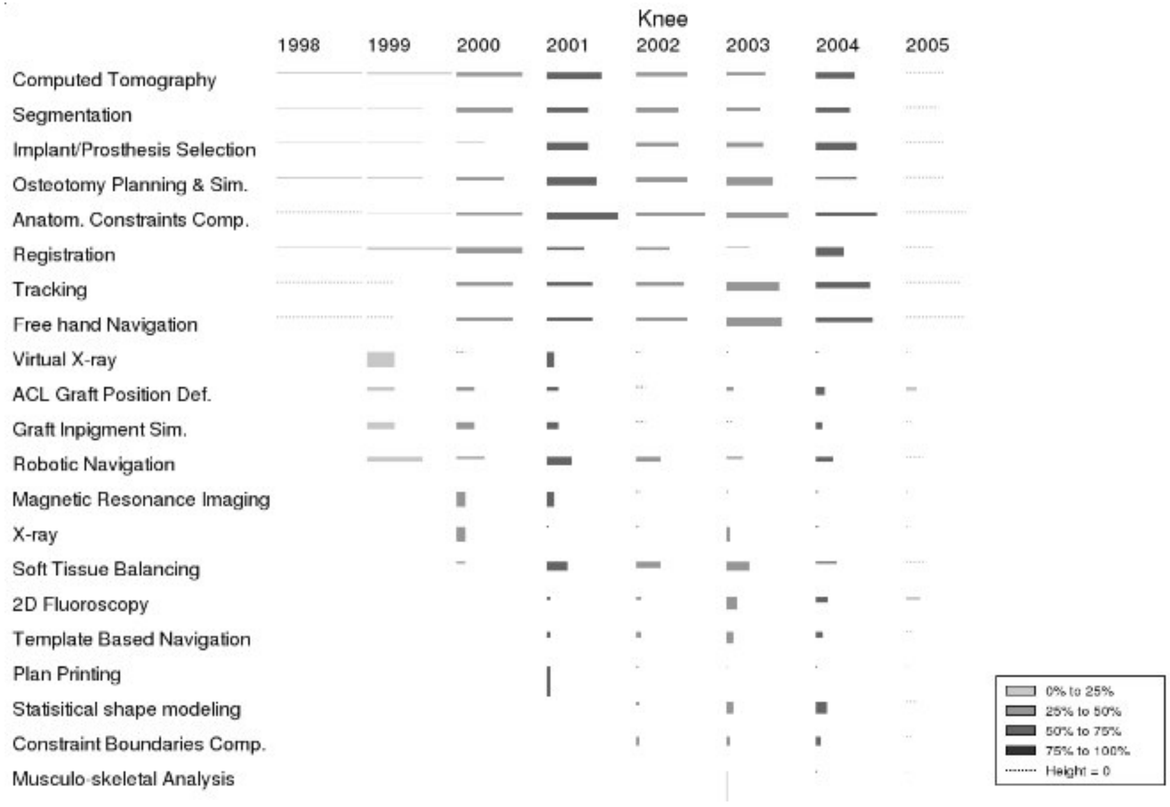


Fig. 6 Evolution Matrix of the Knee Family

5.2.3 Evolution patterns

Table 2 summarizes the evolution patterns identified for each feature; we used a dark gray to mark the features identified as common and a light gray for the variations.

	Spine	Hip/Pelvis	Knee	Trauma
Computed Tomography	Red Giant	Persistent + Red Giant	Red Giant	Red Giant
Tracking	Persistent + Red Giant	Persistent + Red Giant	Red Giant	Persistent + Red Giant
Free hand navigation	Persistent + Red Giant	Persistent + Red Giant	Red Giant	Persistent + Red Giant
2D fluoroscopy	Red Giant	Idle	Idle	Super Nova
Segmentation	White dwarf	Red Giant	Red Giant	Red Giant
Implant/Prosthesis selection	Idle	Super Nova	Red Giant	Idle
Registration	Red Giant	Red Giant	White Dwarf	White Dwarf
Virtual X-ray	Dayfly + Idle	Idle	Idle	Dayfly +Idle
Robotic navigation	Dayfly + Idle	Idle	White Dwarf	▨
Trajectory definition	Red Giant	Idle	▨	White dwarf
Template based navigation	▨	Dayfly + Idle	Idle	Dayfly +Idle
Biomechanical analysis	Dayfly + Idle	Dayfly + Idle	▨	▨
3D fluoroscopy	Dayfly + Idle	▨	▨	Dayfly + Idle
Osteotomy planning and simulation	▨	White Dwarf	Red Giant	▨
Anatomical constraints computation	▨	Super Nova	Red Giant	▨
Segmental alignment	Dayfly + idle	▨	▨	▨
Range of motion simulation	▨	White Dwarf	▨	▨
ACL graft position definition	▨	▨	Idle	▨
Graft impingement simulation	▨	▨	Idle	▨
Magnetic Resonance Imaging	▨	Dayfly + Idle	Dayfly + Idle	▨

X-ray	▨	▨	Idle	▨
Soft tissue balancing	▨	▨	Idle	▨
Plan printing	▨	▨	Dayfly + Idle	▨
Statistical shape estimation	▨	▨	Idle	▨
Constraints boundaries computation	▨	▨	Idle	▨
Musculo-skeletal analysis	▨	▨	Dayfly + Idle	▨
Ultrasound imaging	▨	▨	▨	Idle
Fracture fragment identification	▨	▨	▨	Red Giant or Idle
Virtual fracture reduction	▨	▨	▨	Idle

Table 2. Identified Common and Variable Features

Except for the "fracture fragment identification" feature of the trauma family, which we found has an evolution pattern between Idle and Red Giant we could match each feature evolution to one or two of the previously defined evolution patterns. However, no pulsar pattern was detected. We realized that if a feature is Persistent it is as well a Red Giant and that similarly Dayflies are as well Idle.

Without being Persistent in all families "Computed Tomography" is the only feature apart from "Tracking" and "Free hand navigation" that is a common feature in all families. We then deduced that these three functionalities constitute the common core of the CAOS domain. Sixteen features show only Idle evolution pattern in the families where they are present, and we considered such features as being either variations that did not yet make a breakthrough in the domain or that they have been proposed and abandoned.

Seven of the eight features that are present in more than one family have different evolution patterns from one family to the other, including at least one Red Giant pattern. We use here the particular case where only two sub-families are concerned to explain the conclusions that can be derived. When Red Giant evolution is combined with a Super Nova evolution we can say that the feature was commonly used in a family and was adapted with success to the other. When Red Giant is combined with Idle, either the adaptation failed or did not yet succeed to prove its usability to the related community. Finally, when Red Giant is combined with White Dwarf the feature is commonly used in both families but with a progressively decreasing popularity in the family where the White Dwarf appeared. The last combination of evolution patterns observed is the one of White Dwarf and Idle, which is characteristic of a feature that used to be common in a family and was adapted with minor success to other families.

Moreover, a reader familiar with CAOS technology could expect the evolution of tracking and free-hand navigation to be tightly coupled. Indeed, we can see that they have almost identical evolution patterns in the case of the two presented matrices. We observed the same phenomenon for the "ACL graft position definition" and "Graft impingement simulation" functionalities. Unexpectedly, we realized here that evolution matrices could as well reveal certain kinds of feature dependency.

5.3 Capturing variation

5.3.1 Functional evolution descriptions

In a second run abstracts were collected that do not necessarily contain a system description but rather focus on proposing improvements to a given aspect of already existing systems. Abstracts usually contain motivation for the design and/or use of the described contribution as well as a discussion of its benefits and drawbacks. Because this information usually justifies a change or a novelty in the system we used it to identify the factors of software evolution in the CAOS domain, that is to say the non-functional reasons why CAOS systems were and will be modified and evolved. This second selection of abstracts contained a total of 212 papers describing situations in which only a particular functional aspect of an existing system was modified.

5.3.2 Evolution factors

Reviewing the underlying motivations and claimed benefits of the selected abstracts, we identified the following nine factors as the main factors of CAOS evolution.

I. **Visualization:** CAOS aims at providing surgeons with continuously improving patient specific visualization before and during surgery. For this it provides 3D where initially only 2D visualization was present and provides access to an always increasing amount of anatomical information.

II. **Accuracy and safety:** One of the main goals of CAOS technology is to improve the accuracy with which a surgical procedure can be performed compared to the conventional approach. There are many potential sources of inaccuracy in a CAOS system (image acquisition noise, model generation errors, tracking errors, etc.), and different scenarios are proposed to reduce each type of error. Because undetected computational errors could in the worst case endanger human life, safety and accuracy are tightly coupled in CAOS; consequently, we considered them as being a common evolution factor.

III. **Planning and outcome optimization:** Using computer technology, CAOS helps in defining the optimal planning in order to improve surgery outcome and to reduce long term failures and consequent revision surgeries. To do so CAOS proposes computing various anatomical parameters and performing simulations.

IV. **System handling:** CAOS systems preserve the descriptive and procedural surgical knowledge, but imply a deep change in the operational and interaction aspects of surgeries. To attenuate the resulting steep learning curve and to ease CAOS system handling, research proposes, for example, automating certain tasks or providing more intuitive user interfaces.

V. **Invasiveness:** The invasiveness of a surgical procedure refers to the amount of damage generated to the soft tissue (i.e., skin and muscles) in order to access the surgical site. Because minimal invasiveness results in less pain, scarring, and recovery time for the patient, it is nowadays the tendency to adopt less invasive approaches to perform surgery whenever possible. CAOS follows the same path and proposes various functionalities that support minimally invasive surgical procedures.

VI. **Radiation:** The use of certain imaging modalities implies a consequent radiation exposure for the patient and medical staff. CAOS proposes scenarios where image-based assistance is provided with less radiation.

VII. **Time:** When using CAOS systems additional time is often spent in the operating room and is requested for computational needs (image capture, processing, etc.). The evolution trend is to reduce both burdens.

VIII. **Robustness:** As in the algorithmic and computational domains it is preferable for CAOS to provide functionalities, which are insensitive to the possible variations that occur in clinical conditions. Depending on the considered CAOS functionality this can mean insensitivity to the image quality, to the surgical material used, to the system user, to patient motion or to the individual characteristics of a given pathology (e.g. arthrosis) on the targeted anatomy.

IX. **Cost:** Up to now prices of CAOS systems remain quite high. The aim is to reduce the costs and consequently contribute to improve their acceptance in the surgical field.

5.3.3 Most influential factors estimation

We partitioned our collection of abstracts into seven first level categories. Fig. 7 shows the results of the computation of the evolution factor influence, for each category we colored in red the most influential factors. During this estimation we found only eight abstracts that contained no references to any of the identified evolution factors.

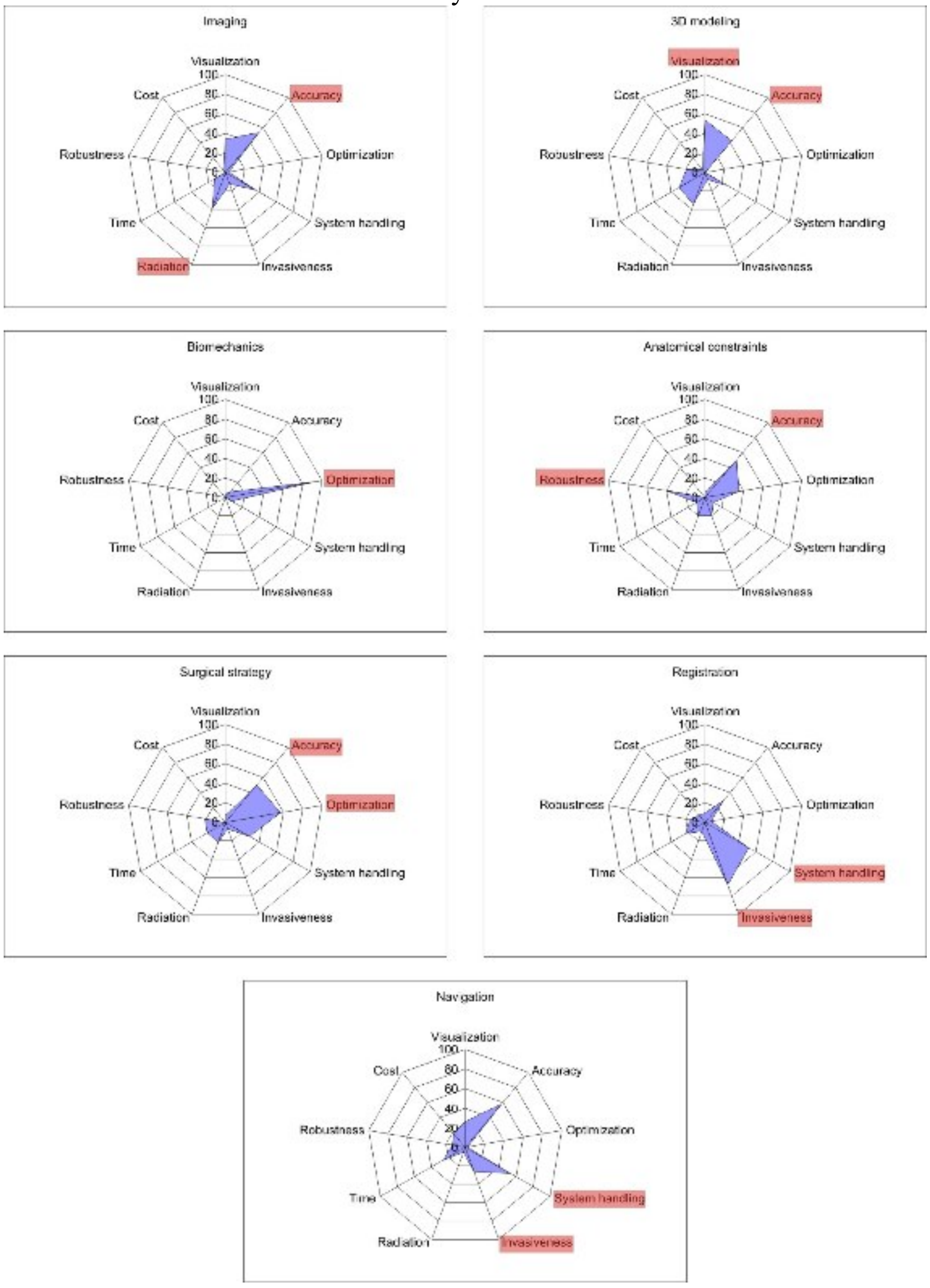


Fig. 7 CAOS most influential evolution factors

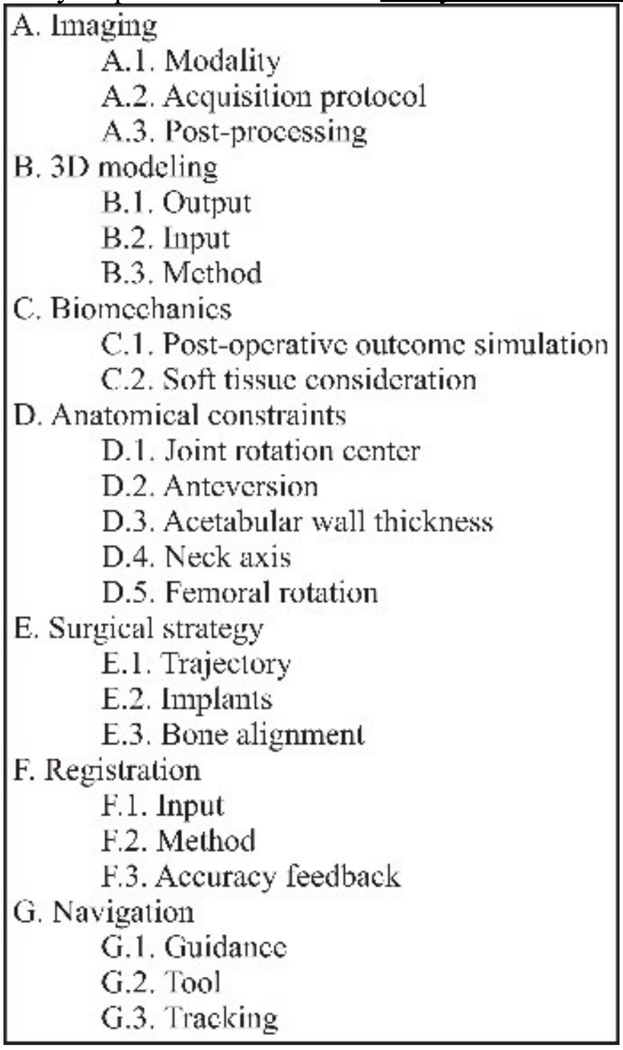
5.3.4 Change scenario taxonomy

We describe here each of the seven taxonomy first level categories highlighting how each of the estimated most influential evolution factors contributes to the functional and technological variations introduced in the related CAOS functionality (the evolution factors are underlined and the resulting subcategories are in bold). The obtained taxonomy is presented in Fig. 8.

Imaging: CAOS aims at enabling <u>minimal radiation exposure</u> of the patient and

medical staff, which implies a permanent investigation on how to use and combine diverse **imaging modalities**. The accurate and precise use of a given imaging modality requires a well defined **acquisition protocol** and/or **calibration procedure** for the means of acquisition. Moreover, to improve the quality and accuracy of the information provided by the acquired images, they sometimes need to be additionally **processed**.
3D modeling: While performing the diagnostics the surgeon has to rely on his or her understanding of the 3D space to interpret the 2D information he or she has access to. One of the claims of CAOS is to ease this challenging task by providing a three-dimensional model of the patient's anatomy that will enhance the surgeon's visualization of the case. Several **methods** were proposed to obtain these models (e.g., segmentation or statistical shape modeling) each of them trying to be always more accurate. What usually vary from one method to another are the type of information it takes as **input** and the nature of the information provided by the **outpu**t model.
Biomechanics: The desire to always obtain better and optimized surgical outcomes led the CAOS community to integrate findings of biomechanics into CAOS. Over the years biomechanical methods such as kinematic analysis and finite element modeling (FEM) have been used to provide surgeons with the possibility to **simulate the post-operative outcome** of the currently planned surgery. Biomechanics contributed as well to one of the major evolutions in CAOS: the idea to no longer rely solely on the patient's skeletal structure but to introduce **soft tissue consideration** in the computer-assisted surgical process. This results, for example, in different ligament balancing functionalities in the knee application family.
Anatomical constraints: An orthopaedic surgical procedure is governed by anatomical constraints **(mechanical axis, anteversion, acetabular wall thickness, etc.)**. In order to allow more accurate surgical procedures CAOS systems propose to automatically compute these parameters whenever possible. A constant feedback on these constraint values improves the guidance offered to surgeons.
Surgical strategy: CAOS planning sub-systems help the surgeon in defining the optimal surgical strategy. For this they provide support in **trajectory** definition, best **implant** selection or virtual **bone alignmen**t in the case of fracture treatment. For a long period these tasks were possible only through user interaction but recent research proposed automating these tasks in order to optimize them.
Registration: In order to be able to guide the surgeon during surgery it is required that CAOS systems establish the mathematical relationship between the local coordinate system of the virtual patient anatomy (generated 3D model) and the one of the surgical object (targeted anatomy to be operated on). This process is called registration or matching. One of the major motivations for the various proposed **methods** to perform this process, apart from their diverse algorithmic approaches, is to reduce the invasiveness required to obtain the data they use as **input**. The second major drawback of registration is that it is a difficult task to perform and understand for the surgeon. Consequently, some effort has been made in order to ease registration handling, e.g., requiring less user interaction or providing intuitive **accuracy feedback**.
Navigation: To help the surgeon achieve his or her surgical plan various **guidance** methods have been proposed such as, for example, individual templates or augmented reality. Although CAOS always tries to be as close as possible to the traditional surgical approach, it sometimes requires modifying or adapting the usual surgical **tools.** Certain navigation approaches need to use positional information about the surgical instruments and anatomical structures. For this there are different **tracking** methods. The goal is

obviously to allow the surgeons to achieve the best accuracy and at the same time to provide systems that only require interaction that easily fits into the clinical routine.



A. Imaging
- A.1. Modality
- A.2. Acquisition protocol
- A.3. Post-processing

B. 3D modeling
- B.1. Output
- B.2. Input
- B.3. Method

C. Biomechanics
- C.1. Post-operative outcome simulation
- C.2. Soft tissue consideration

D. Anatomical constraints
- D.1. Joint rotation center
- D.2. Anteversion
- D.3. Acetabular wall thickness
- D.4. Neck axis
- D.5. Femoral rotation

E. Surgical strategy
- E.1. Trajectory
- E.2. Implants
- E.3. Bone alignment

F. Registration
- F.1. Input
- F.2. Method
- F.3. Accuracy feedback

G. Navigation
- G.1. Guidance
- G.2. Tool
- G.3. Tracking

Fig. 8 Taxonomy of change scenarios

5.4 General CAOS evolution

If we compare the percentages per year of abstracts that contain a system description and of those that can be identified as change scenarios (Fig. 9), we can see that the percentage of system descriptions constantly decreased over the years to reach less than 5% in 2005, while the percentage of change scenarios rapidly grew between 1998 and 2002 and stabilized around 20% afterwards.

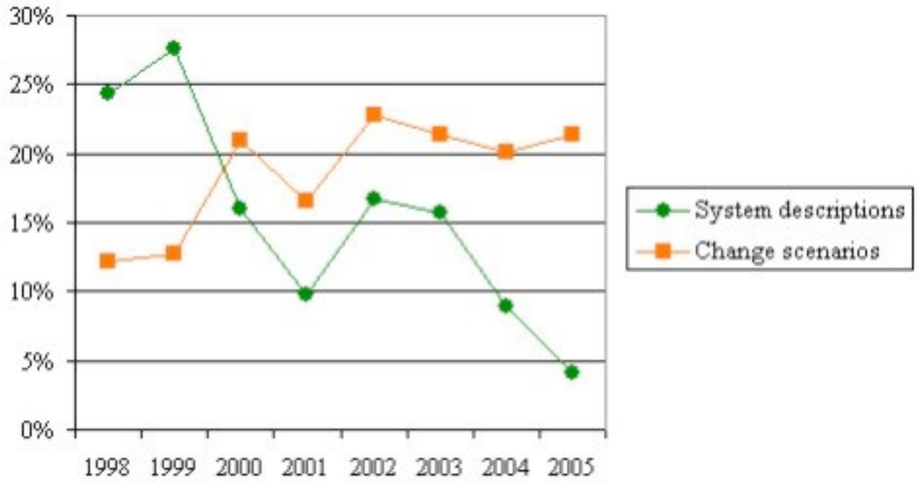


Fig. 9 CAOS Research Evolution

The evolution curves are characteristic of the general evolution of CAOS research. Indeed, rather than investigating the development of a completely new system that will be used for a new surgical procedure, many research groups had focused on improving only a particular aspect of existing CAOS systems. This tendency is as well confirmed by the fact that after 2003 the height of all obtained evolution matrices did not grow, that is to say that after this date the system descriptions no longer contained new features.

6. Efficiency of evolution matrices to support commonality and variability analysis

6.1 Evaluation approach

In order to evaluate how good a given domain analysis method is, the common practice is to evaluate the domain model resulting from the use of this method. To do so, one has to check if it possible to specify pre-existing or proposed systems using the domain analysis outputs. Systems used as inputs in the domain modeling process may be used for validation; but preferably the domain analyst will test using systems not used to develop the model. This last point led us to use an approach borrowed from mathematical modeling, namely holdout validation, to evaluate the effectiveness of evolution matrices in commonality and variability analysis. The principle is the following: observations are randomly selected from the initial data set to form the test data and the remaining data are retained to be used as input to the tested modeling method. Usually up to one third of the initial data is used as test data. After applying the tested method to the training data set one must then estimate the error occurring with the test data [28].

In our particular situation we wanted to evaluate how well the use of evolution matrices could allow us to differentiate common and variable features. Unfortunately, to the best of our knowledge, there is not up to now a measure that could enable the quantitative estimation of our model. We consequently used a qualitative approach to estimate the quality of the obtained domain model. Our evaluation was based on the following assumptions:

- *Assumption 1*: A system description fits well to the domain model if it is composed of a majority of common features plus some variation features.
- *Assumption 2*: After a successful modeling, the number of systems composed only of common features should be minimal.

- *Assumption 3*: A good commonality and variability analysis should not lead to systems composed only of variable features.
- *Assumption 4*: If a feature is missing from the resulting model, it should be included in a minimum of systems descriptions.
- *Assumption 5*: Each common feature should be included in more than 40% of the tested system descriptions. Conversely, variation features should occur in no more tan 40% of the tested system descriptions.

6.2 Evaluation results

6.2.1 General comments

The evaluation was performed once for each sub-family using one third of each sub-family as hold out samples. After applying CompAS we compared the results obtained with and without the complete data set. We noticed that, because of the missing descriptions, in two cases features that were identified as “Persistent + Red Giant” lost their Persistence characteristic but remained Red Giant. The opposite phenomenon was observed as well in one case, that is to say a feature that was initially identified as an Idle only became a “Dayfly + Idle”. Two others complementary effects linked to the hold out sampling were observed. In two cases features that were initially identified as Idle became White Dwarf and in another situation a feature that was a initially Red Giant became a Super Nova. However in these six cases (9.5% of the evaluated features), the observed modifications did not affect the conclusions that were derived concerning the commonality and variability characteristic of these features. Four features that were initially identified as “Dayfly + Idle” and one that was only Idle disappeared from the resulting models. In other words in these five cases (8%) the concerned features were part only of the retrieved systems descriptions. The non detection of such unique features is however not critical; when facing such a situation the domain analyst has to further investigate and find out if it is worthwhile to consider such functionality as a missing reusable variation or to keep it as a truly application specific functionality. Only one feature that we initially identified as Idle became a Red Giant that is to say changed from a variable to a common feature. Finally the only feature which we could not decide whether to classify as a Red Giant or Idle appeared to be a Idle in this evaluation phase.

6.2.2 Domain model evaluation

In this section we present the qualitative evaluation of the obtained model based on the previously mentioned assumptions:

Assumption 1: Across the four sub-families’ evaluation, we found only one system composed mainly of variation features for a small number of common ones.

Assumption 2: For the four sub-families we evaluated, between 20% and 40% of the test set were systems composed only of common features.

Assumption 3: For the four sub-families, no systems were found composed only of variable features.

Assumption 4: Four of the five missing features were required for only one system description and in each case for a different one. The last one was missing for two system descriptions.

Assumption 5: In two cases we found common features present in less than 40% of the test set and in two other cases we found variable feature that was present in more than 40% of the test set. We give below more details on each of the cases:

Case 1 concerns the feature for which we initially had doubt in classifying the followed evolution pattern and which we classified as Idle during our evaluation. This feature appeared to be a variable feature present in 66.7% of the test set. Consequently no decision on whether this feature is common or variable can be taken using CompAS.

Case 2 concerns a feature that we identified with and without the complete set as a White Dwarf. This feature appeared during our evaluation as a variation contained in 44.4% of the test set. Given the fact that we considered a White Dwarf to be a common feature which decreased in popularity to become variable, we can hypothesize that the random sampling retrieved system descriptions that mainly belong to the time period where this feature was considered as common.

Case 3 concerns the previously mentioned feature that was identified as a Red Giant with the complete data set but as a Super Nova during evaluation. It then appeared to be a common feature that belongs to 33.3% of the test set. Although the percentage is below the hypothesized 40% it still remains rather high (one third of the test set). However in this case the assumption has to be rejected.

Finally case 4 concerns a feature identified as a Red Giant with and without the complete set but which appeared as a common feature present in only 22.2% of the test set. To justify this situation one has to use domain knowledge. Indeed, the considered feature, namely computed tomography, is linked to the 2D fluoroscopy feature through an "OR" dependency. In other words, in the trauma sub-family, systems contain either computed tomography feature or the 2D fluoroscopy one. This is actually confirmed by the fact that 2D fluoroscopy feature was evaluated as a common feature that belongs to 88.9% of the test set. In conclusion when two or more common features are dependent they can be contained in a rather small percentage of the considered systems, provided that their respective percentages are complementary (the sum is equal to 100%). Table 3 summarizes the result of the evaluation of the first part of CompAS.

	Spine	Hip	Knee	Trauma
Size of the test set	5	12	18	9
Size of the feature set	13	16	14	21
Number of systems composed mainly of variation features	1	0	0	0
Number of systems composed only of common features	2	4	4	2
Number of systems composed only of variable features	0	0	0	0
How many feature miss in the model	1	0	3	1
What is the number of systems requiring these features	1	NA	2	1
How many common features are included in less than 40% of the test system descriptions	0	0	1	1
How many variable features are included in more than 40% of the test system descriptions	0	0	1	1

Table 3: Results of CompAS Efficiency Evaluation

Because our evaluation assumptions were rejected in a minor number of cases we concluded that evolution matrices are appropriate and efficient tools to support commonality and variability analysis.

7. Discussion

Although software development in research is not constrained by any system documentation process, we demonstrated in this article that relevant information could be extracted from conference proceedings in order to identify commonality and variability based on a quantitative evaluation of software functional evolution. An interesting aspect of the presented results is that they combined users' and developers' perspectives. Indeed, the annual meeting of the International Society for Computer Assisted Orthopaedic Surgery presents research results submitted by surgeons as well as software developers. We actually observed that system functional descriptions could often be found in clinical study reports and reflect the users' perspective at that time. In other cases they were used to describe the current state of the art in articles presenting new developments, thus providing a software developer's perspective. Change scenarios were more often found in technologists' publications. We showed as well that based on an organized review of selected conference abstracts we could identify domain evolution factors. Based on the computation of the dominant evolution factors we could then gain insights into the domain evolution and thereby more easily identify variation points.

The gathering and analysis of our set of data was pretty time consuming, however this is a known aspect of domain analysis [29, 30]. And we believe, as is usually advocated in the domain, that the obtained results are worth the time invested since they will save time in the future.

Our interpretation of the relation between the obtained evolution matrices and the commonality, variability, and dependency of the features could not have been possible without our *a priori* knowledge of the domain. In other words such an approach requires good knowledge of the investigated domain and/or a close collaboration with domain experts. The proposed approach presents, moreover, a limitation related to the data collection. The system descriptions used to compute the evolution matrices do not follow any template or format and are therefore subjective. Some descriptions might therefore be incomplete, simply because the omitted functionalities were not relevant to the treated subject; they were implicit for the audience, or became standard and therefore were not mentioned anymore. Based on our knowledge of CAOS systems, we have actually sometimes extrapolated the presence of certain functionalities in order to keep the system descriptions coherent.

Because of the previously mentioned general evolution of CAOS research, we had only a very small number of system descriptions for the years 2004 and 2005. This evolution tendency can bias the interpretation for the evolution matrix. If we consider, for example, the "3D fluoroscopy" feature which is interpreted as a feature with relatively low popularity (Idle pattern), the matrix does not allow one to detect that this functionality was indeed investigated intensively as a stand-alone feature. This growing popularity is indeed confirmed by the numerous change scenarios (17) related to "3D fluoroscopy" that we collected between the years 2001 and 2004.

We have seen that feature dependencies, such as constraints describing which features require the presence of one or more features, imply identical evolution patterns for the respective features. Other feature dependencies exist but could neither be detected nor integrated with our approach. If we consider, for example, introducing a new imaging modality, we know that it may require a new 3D modeling approach to be defined. For these reasons we insist here on the fact that the presented work, in its current status, can only be used as a tool to support and strengthen one part of domain analysis (i.e. commonality and variability analysis). The lack of the critical dimension that constitutes dependency handling prevents this approach from being considered as a complete domain analysis method. We therefore envision evolving our taxonomy into an ontology so that we can model these dependencies. Ontologies enable the definition of concepts (change scenario, surgical procedure, medical image, etc.), their attributes and the relationships among them [31]. An ontology of CAOS change scenarios would permit one to define a human and computer readable consensual description of CAOS system variability that could be shared and reused.

The high number of identified Idle patterns demonstrates that CAOS is a highly evolving domain where numerous functionalities are still investigated and proposed.

We have seen as well that functionalities are frequently adapted from one sub-family to the other. Moreover, there is a tendency in the CAOS community to focus on developing new functionalities rather than taking a systems implementation approach to consolidate and integrate existing results. All these characteristics reinforce our initial intuition that a software reuse philosophy could be of interest for the domain. In a component-based environment one could test, integrate, and adapt a specific contribution without having to implement a completely new system but rather by combining the newly provided features with others to develop a more functional system.
As demonstrated by some of our results, until now, CAOS research has largely addressed major issues such as safety and accuracy. Other issues like cost reduction have not yet been satisfactorily investigated. Initial studies in industrial settings suggest that component-based application engineering results in an improvement of programmer productivity, a reduction of time-to-market, and a decrease of maintenance costs. Consequently, we believe that an academic investigation of the use of component-based development for CAOS systems could open the door to less costly industrial production of CAOS systems.

8. Future work

The presented overview of commonality and variability in CAOS provides an abstract representation of the domain. The previously mentioned lacking information about feature dependencies as well as other information such as illegal feature combinations and default settings constitutes what is called the knowledge configuration of the domain. Combining our domain model with configuration knowledge would enable one to configure any concrete CAOS applications. This transition from a high level domain model to a concrete system description can nowadays be performed either manually or automatically. In the manual approach the developer, based on his or her understanding of the domain model, matches a system description expressed in common language to a computer readable description of the system. In generative programming, which proposes methods to automate the process, the developer uses a domain specific language (DSL) to describe applications and the transition to concrete implementations is performed automatically [6].
We will investigate if the use of an ontology as a means for domain modeling could allow us to encode the knowledge configuration. We could then provide a means to ease and automate not all the transition process but only the transition from the high level domain representation to a computer readable system description. Indeed concept of relationships, axioms, and rules that are part of ontological engineering could be used to define the knowledge configuration. Moreover, the consensual aspect of the definition of ontology terminology could lead to an application descriptor whose vocabulary would be accessible to all the actors of the domain. The dynamic aspect of ontology that is to say the fact that they are made to be extended and modified could ensure an always up-to-date representation of the domain. Finally, ontologies are not only human readable but as well computer readable, so we could then investigate the development of an ontology based tool that will allow us to provide computer support in application description.
We could imagine that ultimately the functional aspect of our domain model could be used to provide users with a computer-based catalog of the available features that could help in configuring applications but as well in informing about the possible functional improvements of a given application. While the technological aspect could be used as a source of information on how to improve the non-functional aspect of a system based on

the different existing technological approaches for a given feature.

Acknowledgement: This work was supported by the Swiss National Center of Competence in Research on "Computer Aided and Image Guided Medical Intervention (CO-ME)".

References

[1] A.M.M.A. Mohsen, T.J. Cain, M.R.K. Karpinski, K.P. Sherman, F.R. Howell, R., Phillips et al., The basic orthopaedic principle and the non-invasive intelligent orthopaedic guide concept, in Proceedings of the 2nd International Workshop of Mechatronics in Medicine and Surgery (Medimec), Bristol, UK, 1995.

[2] F. Langlotz, L.P. Nolte, Computer-assisted orthopaedic surgery from theory to the operating room, Techniques in Orthopaedics 18 (2003) 140-148.

[3] L.P Nolte, T. Beutler, Basic principles of CAOS, Injury 35 (2004) SA6-SA16.

[4] C. Szyperski, Component Software - Beyond Object-Oriented Programming, Addison-Wesley, 1998.

[5] D.L. Parnas, On the design and development of program families, IEEE Transactions on Software Engineering SE-2 (1976) 1-9.

[6] K. Czarnecki, U.W. Eisenecker, Generative Programming: Methods, Tools, and Applications, Addison-Wesley, 2000.

[7] D. Draheim, L. Pekacki, Process-Centric Analytical Processing of Version Control Data, in Proceedings of the International Workshop on Principles of Software Evolution (IWPSE), Helsinki, Finland, 2003.

[8] H. Gall, M. Jazayeri, J. Krajewski, CVS release history data for detecting logical couplings, in Proceedings of the International Workshop on Principles of Software Evolution (IWPSE), Helsinki, Finland, 2003.

[9] M. Fischer, H. Gall, Visualizing Feature Evolution of Large Scale Software based on Problem and Modification Report Data, Journal of software maintenance and evolution : Research and Practice 16 (2004) 385-403.

[10] A.I. Anton, C. Potts, Functional Paleontology: System evolution as the User sees, in Proceedings of the 23rd Internatinal Conference on Software Engineering (ICSE), Toronto, Canada, 2001.

[11] M. Lanza, The evolution matrix: recovering software evolution using software visualization techniques, in Proceedings of the International Workshop on Principles of Software Evolution, Vienna, Austria, 2001.

[12] Cambridge dictionaries online. dictionary.cambridge.org [April 13th, 2006]

[13] G. Arango, Domain analysis methods, in: W. Schäfer, R. Prieto-Díaz, M. Matsumoto (Eds.) Software Reusability, Ellis Horwood, New York, NY, 1994, pp 17-49.

[14] S. Wartik, R. Prieto-Díaz, Criteria for comparing domain analysis approaches, International Journal of Software Engineering and Knowledge Engineering 2 (1992) 403-431.

[15] K. Kang, S. Cohen, J. Hess, W. Nowak, S. Peterson, Feature-Oriented Domain Analysis (FODA) Feasibility Study, Technical report CMU/SEI-90-TR-21, Software Engineering Institute, Carnegie Mellon University, Pittsburg PA, 1990, www.sei.cmu.edu/publications/documents/90.reports/90.tr.021.html [April 13th, 2006]

[16] Software Technology for Adaptable Reliable Systems (STARS). Organization Domain Modeling (ODM) Guidebook Version 2.0, STARS-VC-A025/001/00, Lockheed Martin Tactical Defense Systems, Manassas VA, 1996. www.cs.bu.edu/faculty/allemang/Papers/ODM.pdf [April 13th, 2006]

[17] W. Tracz, L. Coglianese, P. Young, Domain-specific Software Architecture engineering Process Guidelines, ADAGE-IBM-92-02, IBM Corporation, Federal Sector Division, Owego NY, 1993, citeseer.ist.psu.edu/tracz93domainspecific.html [April 13th, 2006]
[18] J. Bayer, D. Muthig, and T. Widen, Customizable Domain Analysis, in Proceedings of the 1st International Symposium on Generative and Component-Based Software (GCSE) Engineering , Erfurt, Germany, 1999
[19] R. Holibaugh, Joint Integrated Avionics Working Group (JIAWG) Object Oriented Domain Analysis Method (JODA), Version 1.3, Technical Report CMU/SEI-92-SR-3, Software Engineering Institute, Carnegie Mellon University, Pittsburg PA, 1993, www.sei.cmu.edu/pub/documents/92.reports/pdf/sr03.92.pdf [April 13th, 2006]
[20] H. Gomaa, An Object-Oriented Domain Analysis and Modeling Method For Software Reuse, in Proceedings of 25th the Hawaii International Conference on System Sciences, Kauai HI, USA, 1992
[21] W. Vitaletti, E. Guerrieri, Domain Analysis within the ISEC RAPID Center, in Proceedings of the 8th Annual National Conference on ADA Technology, Atlanta GA, USA, 1990
[22] W. Frakes, R. Prieto-Diaz, C. Fox, DARE: Domain analysis and reuse environment, Annals of Software Engineering 5 (1998) 125-141.
[23] J.M. Neighbors, Draco: a method for engineering reusable software components, in: T.J. Biggerstaff, A. Perlis (Eds.), Software Reusability, Vol. 1, Addison-Wesley/ACM Press, 1989.
[24] Y. V. Srinivas, R. Jüllig, SPECWARE: Formal Support for Composing Software, in Proceedings of the International Conference on the Mathematics of Program Construction, Kloster Irsee, Germany, 1995
[25] D.R. Smith, KIDS: A semi-automatic program development system, IEEE Transactions on Software Engineering 16 (1999), 1024-1043.
[26] M. Lanza, S. Ducasse, Polymetric views – A lightweight visual approach to reverse engineering, in IEEE Transactions on Software Engineering 20 (2003) 782-795.
[27] Trolltech Qt 3.3 whitepaper,
www.trolltech.com/pdf/whitepapers/qt33-whitepaper-a4.pdf [April 13th, 2006]
[28] R. Kohavi, A study of cross-validation and bootstrap for accuracy estimation and model selection, in Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI), Montréal, Québec, Canada ,1995
[29] A. Fantechi, S. Gnesi, I. John, G. Lami, J. Dörr, Elicitation of Use Cases for Product Lines, in Proceedings of the Fifth International Workshop on Product Family Engineering (EPF), Siena, Italy, 2003
[30] K. Lee, K.C. Kang, J. Lee, Concepts and Guidelines of Feature Modeling for Product Line Software Engineering, in Proceedings of the Seventh International Conference on Software Reuse: Methods, Techniques, and Tools, Austin, USA, 2002
[31] B. Chandrasekaran, R. Josephson, R. Benjamins. What Are Ontologies, and Why Do We Need Them, in IEEE Intelligent Systems 14 (1999) 20-26.