Alma Mater Studiorum - Università di Bologna

FACOLTA' DI SCIENZE MATEMATICHE, FISICHE E NATURALI
Corso di LAUREA TRIENNALE in Informatica

LA CERTIFICAZIONE DEL SOFTWARE MEDICALE: CASI DI STUDIO

Tesi di Laurea
Di
FEDERICA CAPELLINI

Relatore
Chiar.mo Prof.
COSIMO LANEVE

Sessione I

Anno Accademico 2009/2010

1 LA DIRETTATIVA EUROPEA 2007/47/EC

Da una analisi condotta negli Stati Uniti è emerso che il 7% dei richiami del mercato su "*medical devices"*, tra il 1992 e il 1998, erano attribuibili a difetti del software. Sicuramente basterebbe come affermazione per caratterizzare l'importanza cha va assumendo, al pari passo con l'aumento del software nella applicazioni mediche, la verifica del software medicale. Ma lo stesso aspetto di verifica viene in questo periodo ribadito dal legislatore, con l'entrata in vigore dal 21 marzo del 2010 della Direttiva Europea 2007/47/EC che modifica la precedente direttiva 93/42 CEE del 14 giugno 1993.

La direttiva 2007/47/EC, emanata il 5 settembre 2007, rappresenta il risultato di un importante lavoro di mediazione fra gli stati membri dell'Unione e introduce alcuni importanti modifiche alla normativa precedente, fra le quali vale la pena di citare:

- *Prodotti" borderline" e classificazione dei dispositivi medici*
- *Organizzazione e vigilanza post marketing*
- *Nuovi requisiti per dati clinici e valutazione clinica*
- *Convalida del software*

In questo elaborato ci occuperemo principalmente di illustrare le introduzioni normative relative alla convalida del software, esaminandone i punti che sembrano maggiormente significativi. A tal fine utilizzeremo un caso effettivo di software medicale stand-alone. Il nostro intendimento è infatti che solo declinando la normativa su casi reali si possa comprendere l'impatto che essa ha introdotto e quali siano i punti di contatto con gli aspetti di ingegnerizzazione del software proposti negli ultimi anni. Infatti parlare di qualità del software non è certo solo una esigenza legata alla situazione legislativa in atto: è già da molto tempo che la comunità scientifica si interroga relativamente a queste tematiche.

Inizieremo facendo una breve esposizione dei passaggi più significativi della normativa europea 2007/47/EC, tralasciando le parti che non hanno

attinenza diretta alla certificazione del software, con l'intenzione di fornire in dettaglio le disposizioni di legge in materia.

Prima però di esaminare compiutamente il testo ci interessa ribadire alcuni passaggi chiave della normativa, che, parlando di software, ci sembrano meritare un approfondimento.

Dalla disamina degli allegati emergono molti punti sui quali ci si interroga anche quando si parla di qualità del software in ambito scientifico, e la norma 2007/47/EC ne fa esplicito riferimento:

"*Per i dispositivi che incorporano un software o costituiscono in sé un software medico, il software è convalidato secondo lo stato dell'arte, tenendo conto dei principi del ciclo di vita dello sviluppo, della gestione dei rischi, della validazione e della verifica"* (All I)

Di più, in ambito informatico si riconosce che, se da una parte non ci sono metodi per garantire la correttezza al 100% per ogni tipo di software, è comunque necessario cercare di perseguire la correttezza utilizzando tre principi base:

1. risk management
2. quality management
3. software engeneering

Nelle normativa troviamo declinati questi principi. Infatti essa raccomanda:

- I dispositivi devono fornire le prestazioni loro assegnate dal fabbricante ed essere progettati, fabbricati e condizionati …(All I)
- I dispositivi con funzione di misura devono essere progettati e fabbricati in modo tale da fornire una costanza e precisione di misura adeguate… (All I)
- La documentazione tecnica deve contenere:
    - una descrizione generale del prodotto, comprese le varianti previste e gli usi cui è destinato;
    - gli schemi di progettazione e i metodi di fabbricazione, gli schemi delle parti, dei pezzi, dei circuiti, ecc.;

- la descrizione e le spiegazioni necessarie per la comprensione degli schemi summenzionati e del funzionamento del prodotto;
- i risultati dell'analisi dei rischi e ..una descrizione delle soluzioni adottate..
- i risultati dei calcoli di progettazione, dei controlli svolti, ecc. (All VII)

Un aspetto particolare assume l'analisi dei rischi, che in ambito medicale significa:

- eliminare o ridurre i rischi nella misura del possibile (integrazione della sicurezza nella progettazione e nella costruzione del dispositivo) (All I);
- se del caso adottare le opportune misure di protezione nei confronti dei rischi (All I);
- informare gli utilizzatori dei rischi residui dovuti a un qualsiasi difetto delle misure di protezione adottate (All I);
- Tutti i dispositivi devono contenere nell'imballaggio le istruzioni per l'uso (All I).

Un altro aspetto sempre delicato nel software è rappresentato dalla documentazione che ne accompagna l'istallazione e l'utilizzo. Fra gli aspetti da verificare nel conseguimento della certificazione è compreso anche quello di fornire istruzioni per l'uso e etichettature per il corretto utilizzo dello strumento medicale. (All I)

D'altra parte è noto che il ciclo di vita del software non termina con il rilascio, ma prosegue con :

- il mantenimento dello stesso
- la soluzione degli eventuali errori

Possiamo leggere in questo contesto le ulteriori raccomandazioni dove si raccomanda:

- Il fabbricante istituisce e aggiorna una procedura sistematica di valutazione dell'esperienza acquisita sui dispositivi nella fase successiva alla produzione.. (All VII);
- Informa le autorità competenti degli incidenti seguenti, non appena egli ne venga a conoscenza (All VII).

Dall'allegato X invece si richiedono validazioni cliniche che, nel caso di software stand-alone, tratterà la correttezza del software stesso.

1.1 Il testo della normativa

Una premessa: il testo di riferimento seguito è strutturato in maniera da indicare con lettere le revisioni che si sono succedute a partire dalla prima stesura della normativa, in modo da evidenziare le novità introdotte (lettera M5) dall'ultima versione, alcune delle quali molto significative per il tema che stiamo trattando.

Di seguito denomineremo:

►B DIRETTIVA 93/42/CEE DEL CONSIGLIO del 14 giugno 1993 concernente i dispositivi medici (GU L 169 del 12.7.1993, pag. 1)

Modificata da:

►**M1** Direttiva 98/79/CE del Parlamento europeo e del Consiglio del 27 ottobre 1998

►**M2** Direttiva 2000/70/CE del Parlamento europeo e del Consiglio del 16 novembre 2000

►**M3** Direttiva 2001/104/CE del Parlamento europeo e del Consiglio del 7 dicembre 2001

►**M4** Regolamento (CE) n. 1882/2003 del Parlamento europeo e del Consiglio del 29 settembre 2003

►**M5** Direttiva 2007/47/CE del Parlamento europeo e del Consiglio del 5 settembre 2007

Passiamo ora alla disamina del testo.

1.2 Articolo 1

▼*B 1. La presente direttiva si applica ai dispositivi medici e ai relativi accessori.*

troviamo subito un passo dove si definisce ‘dispositivo medico’ anche il software:

2. Ai fini della presente direttiva s'intende per:

a) ►***M5 «dispositivo medico»****: qualunque strumento, apparecchio, impianto****, software****, sostanza o altro prodotto, utilizzato da solo o in combinazione,* ***compreso il software destinato dal fabbricante ad essere impiegato specificamente con finalità diagnostiche e/o terapeutiche e necessario al corretto funzionamento del dispositivo****, destinato dal fabbricante ad essere impiegato sull’uomo a fini di:* ◄

— diagnosi, prevenzione, controllo, terapia o attenuazione di una malattia;

— diagnosi, controllo, terapia, attenuazione o compensazione di una ferita o di un handicap;

— studio, sostituzione o modifica dell'anatomia o di un processo fisiologico;

Un’altro aspetto su cui si sofferma la normativa è l’indagine clinica che ritroveremo come requisito dell’allegato X, di cui viene data una nuova definizione;

▼*M5*

k) «dati clinici»: informazioni sulla sicurezza e/o sulle prestazioni ricavate dall’impiego di un dispositivo. I dati clinici provengono dalle seguenti fonti:

— indagini cliniche relative al dispositivo in questione,

— indagini cliniche o altri studi pubblicati nella letteratura scientifica, relativi a un dispositivo analogo di cui è dimostrabile l’equivalenza al dispositivo in questione,

— relazioni pubblicate e/o non pubblicate su altre pratiche cliniche relative al dispositivo in questione o a un dispositivo analogo di cui è dimostrabile l’equivalenza al dispositivo in questione;

1.3 Articolo 2

Dove si stabilisce la necessità della certificazione.

▼*M1*

Gli Stati membri adottano le disposizioni necessarie affinché i dispostivi possano essere immessi in commercio e/o messi in servizio unicamente qualora rispondano alle condizioni prescritte dalla presente direttiva, siano correttamente forniti e installati, siano oggetto di un'adeguata manutenzione e siano utilizzati in conformità della loro destinazione.

1.4 Articolo 9

Un'altra novità sono i criteri di classificazione:

1. I dispositivi sono suddivisi nelle seguenti classi: classi I, IIa, IIb e III.

Ci focalizzeremo solo sulle classi I e IIa, che sono quelle previste per il software stand-alone.

Le regole vengono esplicitate nell'allegato IX:

1.4. **Dispositivo medico attivo**

Dispositivo medico dipendente, per il suo funzionamento, da una fonte di energia elettrica o di altro tipo di energia, diversa da quella generata direttamente dal corpo umano o dalla gravità e che agisce convertendo tale energia. Un dispositivo medico destinato a trasmettere, senza modificazioni di rilievo, l'energia, le sostanze o altri elementi tra un dispositivo medico attivo e il paziente non è considerato un dispositivo medico attivo. ►***M5*** *Il* ***software indipendente (stand-alone) è considerato un dispositivo medico attivo*** ◄.

Seguono le regole di applicazione per determinare la classe di appartenenza; la prima cosa di cui tenere conto per determinate la classe è la destinazione d'uso:

▼B

2.1. L'applicazione delle regole di classificazione deve basarsi sulla destinazione dei dispositivi.

2.2. Se un dispositivo è destinato ad essere utilizzato in combinazione con un altro dispositivo, le regole di classificazione devono applicarsi

separatamente a ciascun dispositivo. Gli accessori sono classificati separatamente dal dispositivo con cui sono impiegati.

2.3. ***Il software destinato a far funzionare un dispositivo o ad influenzarne l'uso rientra automaticamente nella stessa classe del dispositivo.***

2.4. Se un dispositivo non è destinato ad essere utilizzato esclusivamente o principalmente in una determinata parte del corpo, esso deve essere considerato e classificato in base all'utilizzazione più critica specificata.

2.5. Se ad un dispositivo si applicano più regole, tenuto conto delle prestazioni che gli sono assegnate dal fabbricante, si applicano le regole più rigorose che portano alla classificazione più elevata.

Tralasciando i dispositivi invasivi, che non ci competono, nella classificazione di quelli non invasivi si deve seguire questo schema:

▼B

1.1. **Regola 1**

Tutti i dispositivi non invasivi rientrano nella classe I, a meno che non sia d'applicazione una delle regole seguenti.

3.1. **Regola 9**

Tutti i dispositivi attivi terapeutici destinati a rilasciare o a scambiare energia rientrano nella classe IIa a meno che le loro caratteristiche siano tali da permettere loro di rilasciare energia al corpo umano o scambiare energia con il corpo umano in forma potenzialmente pericolosa, tenuto conto della natura, della densità e della parte in cui è applicata l'energia, nel qual caso essi rientrano nella classe IIb.

Tutti i dispositivi attivi destinati a controllare o a sorvegliare le prestazioni di dispositivi attivi terapeutici appartenenti alla classe IIb, o destinati ad influenzare direttamente la prestazione di tali dispositivi, rientrano nella classe IIb.

3.2. **Regola 10**

I dispositivi attivi destinati alla diagnosi rientrano nella classe IIa se:

▼*B*

— sono destinati a rilasciare energia che sarà assorbita dal corpo umano, ad esclusione dei dispositivi utilizzati per illuminare il corpo del paziente nello spettro visibile;

— sono destinati a visualizzare in vivo la distribuzione di radiofarmaci in vivo;

— sono destinati a consentire una diagnosi diretta o un controllo dei processi fisiologici vitali, a meno che siano specificamente destinati a controllare i parametri fisiologici vitali, ove la natura delle variazioni è tale da poter creare un pericolo immediato per il paziente, per esempio le variazioni delle funzioni cardiache, della respirazione o dell'attività del sistema nervoso centrale, nel qual caso essi rientrano nella classe IIb.

I dispositivi attivi destinati ad emettere radiazioni ionizzanti e destinati alla diagnosi, alla radioterapia o alla radiologia d'intervento, compresi i dispositivi che li controllano o che influenzano direttamente la loro prestazione, rientrano nella classe IIb.

3.3. **Regola 12**

Tutti gli altri dispositivi attivi rientrano nella classe I.

4.4. **Regola 16**

►***M5*** *I dispositivi* ◄ *destinati specificamente a registrare le immagini diagnostiche ottenute con raggi X rientrano nella classe IIa.*

1.5 Articolo 3

Una volta classificato il dispositivo:

▼*B*

Requisiti essenziali

I dispositivi devono soddisfare i pertinenti requisiti essenziali prescritti nell'allegato I in considerazione della loro destinazione.

{...}

Nell’allegato I vengono formulati i requisiti di base, in particolare legati alla disamina dei rischi legati all’utilizzo del dispositivo stesso e alle specifiche di costruzione, prescindere dal tipo di iter da utilizzare per ottenere la certificazione, e si istituisce l’obbligo di valutazione clinica secondo le disposizioni dell’allegato X:

▼*M5*

1. I dispositivi devono essere progettati e fabbricati in modo che la loro utilizzazione, se avviene alle condizioni e per gli usi previsti, non comprometta lo stato clinico o la sicurezza dei pazienti, né la sicurezza e la salute degli utilizzatori ed eventualmente di terzi, fermo restando che gli eventuali rischi associati all'uso previsto debbono essere di livello accettabile in rapporto ai benefici apportati al paziente e compatibili con un elevato livello di protezione della salute e della sicurezza.

Ciò comporta:

— la riduzione, per quanto possibile, dei rischi di errore nell'utilizzazione determinato dalle caratteristiche ergonomiche del dispositivo e dall'ambiente in cui è previsto che il dispositivo sia usato (progettazione per la sicurezza del paziente), e

— la considerazione del livello della conoscenza tecnica, dell'esperienza, dell'istruzione e della formazione nonché, a seconda dei casi, delle condizioni mediche e fisiche degli utilizzatori cui il dispositivo è destinato (progettazione per utilizzatori comuni, professionisti, disabili o altro).

▼*B*

2. Le soluzioni adottate dal fabbricante per la progettazione e la costruzione dei dispositivi devono attenersi a principi di rispetto della sicurezza, tenendo conto dello stato di progresso tecnologico generalmente riconosciuto.

Per la scelta delle soluzioni più opportune il fabbricante deve applicare i seguenti principi, nell'ordine indicato:

— eliminare o ridurre i rischi nella misura del possibile (integrazione della sicurezza nella progettazione e nella costruzione del dispositivo);

— se del caso adottare le opportune misure di protezione nei confronti dei rischi che non possono essere eliminati eventualmente mediante segnali di allarme;

— informare gli utilizzatori dei rischi residui dovuti a un qualsiasi difetto delle misure di protezione adottate.

3. I dispositivi devono fornire le prestazioni loro assegnate dal fabbricante ed essere progettati, fabbricati e condizionati in modo tale da poter espletare una o più delle funzioni di cui all'articolo 1, paragrafo 2, lettera a), quali specificate dal fabbricante.

6. Qualsiasi effetto collaterale o comunque negativo deve costituire un rischio accettabile rispetto alle prestazioni previste.

▼*M5*

6 **bis**. *La dimostrazione della conformità con i requisiti essenziali deve comprendere una valutazione clinica a norma dell'allegato X.*

▼*B*

{...}

10. Dispositivi con funzione di misura

10.1. I dispositivi con funzione di misura devono essere progettati e fabbricati in modo tale da fornire una costanza e precisione di misura adeguate, entro appropriati limiti di precisione, tenuto conto della destinazione del dispositivo. Detti limiti sono specificati dal fabbricante.

10.2. La scala di misura, di controllo e di indicazione deve essere progettata sulla base di principi ergonomici tenendo conto della destinazione del dispositivo.

▼*B*

{...}

12. Requisiti per i dispositivi medici collegati o dotati di una fonte di energia

12.1. I dispositivi che contengono sistemi elettronici programmabili devono essere progettati in modo tale da garantire la riproducibilità, l'affidabilità e le prestazioni di questi sistemi conformemente all'uso cui sono destinati. In caso di condizione di primo guasto (del sistema) dovranno essere previsti mezzi adeguati per eliminare o ridurre il più possibile i rischi che ne derivano.

▼*M5*

12.1 **bis.** ***Per i dispositivi che incorporano un software o costituiscono in sé un software medico, il software è convalidato secondo lo stato dell'arte,***

tenendo conto dei principi del ciclo di vita dello sviluppo, della gestione dei rischi, della validazione e della verifica.

{...}

▼*B*

p) il grado di precisione indicato per i dispositivi di misura;

Nell'allegato è presente anche una dettagliata disamina delle caratteristiche che devono soddisfare le istruzioni da corredare obbligatoriamente al dispositivo, che tralasciamo per la scarsa attinenza alla nostra analisi.

1.6 Articolo 10

In questo articolo si ribadisce la necessità di monitorare il dispositivo anche nella fase di post-vendita.

▼B

Informazioni riguardanti incidenti verificatesi dopo l'immissione in commercio

1. Gli Stati membri prendono i provvedimenti necessari affinché i dati loro comunicati secondo il disposto della presente direttiva e riguardanti gli incidenti di seguito elencati che hanno coinvolto un dispositivo appartenente ad una delle classi I, IIa, IIb o III siano classificati e valutati a livello centrale:

{...}

1.7 Articolo 11

Dove si forniscono le regole per determinare la Valutazione della conformità. Ci limiteremo, come per la classificazione, alle regole per i dispositivi di tipo I e IIa.

▼B

{...}

2. Per i dispositivi appartenenti alla classe IIa, ad esclusione dei dispositivi su misura e dei dispositivi destinati ad indagini cliniche, il fabbricante deve, ai fini dell'apposizione della marcatura CE, seguire la

procedura per la dichiarazione di conformità CE di cui all'allegato VII, unitamente:

a) alla procedura relativa alla verifica CE di cui all'allegato IV, oppure

b) alla procedura relativa alla dichiarazione di conformità CE (garanzia di qualità della produzione) di cui all'allegato V, oppure

▼*B*

c) alla procedura relativa alla dichiarazione di conformità CE (garanzia di qualità del prodotto) di cui all'allegato VI.

In sostituzione di tali procedure, il fabbricante può seguire la procedura prevista al paragrafo 3, lettera a).

3. Per i dispositivi appartenenti alla classe IIb, diversi dai dispositivi su misura e dai dispositivi destinati ad indagini cliniche, il fabbricante deve seguire, ai fini dell'apposizione della marcatura CE:

a) la procedura relativa alla dichiarazione di conformità CE (sistema completo di garanzia di qualità) di cui all'allegato II; in tal caso non si applica il punto 4 dell'allegato II, oppure

{...}

5. Per i dispositivi appartenenti alla classe I, ad esclusione dei dispositivi su misura e di quelli destinati ad indagini cliniche, il fabbricante ai fini dell'apposizione della marcatura CE, si attiene alla procedura prevista all'allegato VII e redige, prima dell'immissione in commercio, la dichiarazione di conformità CE richiesta.

{...}

7. Nel procedimento di valutazione della conformità del dispositivo, il fabbricante e/o l'organismo notificato tengono conto di tutti i risultati disponibili delle operazioni di valutazione e di verifica eventualmente svolte, secondo il disposto della presente direttiva, in una fase intermedia della fabbricazione.

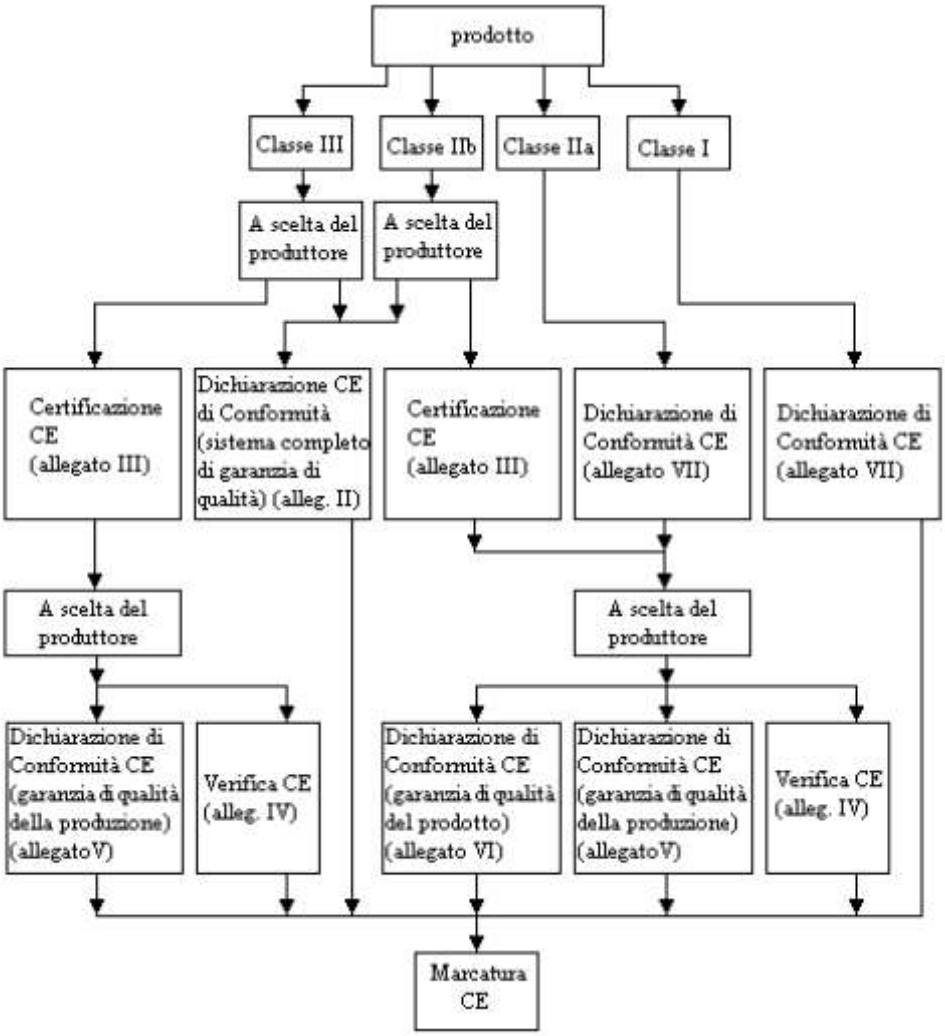
8. Il fabbricante può incaricare il mandatario ►***M5*** _____ ◄ *di avviare i procedimento previsti agli allegati III, IV, VII e VIII.*

9. Se il procedimento di valutazione della conformità presuppone l'intervento di un organismo notificato, il fabbricante o il suo mandatario

►***M5*** _____ ◄*, può rivolgersi ad un organismo di sua scelta nell'ambito delle competenze per le quali l'organismo stesso è stato notificato.{...}*

1.8 ALLEGATO VII

Dove si riportano i requisiti principali per ottenere la DICHIARAZIONE DI CONFORMITÀ CE, di cui riportiamo di seguito uno schema riassuntivo.



▼*M5*

1. Con la dichiarazione di conformità CE il fabbricante o il suo mandatario che soddisfino gli obblighi previsti al punto 2 e, nel caso di prodotti immessi in commercio in confezione sterile o di strumenti di misura, quelli previsti al punto 5, garantiscono e dichiarano che i prodotti in questione soddisfano le disposizioni applicabili della presente direttiva.

2. Il fabbricante predispone la documentazione tecnica descritta al punto 3. {...}

▼*B*

3. La documentazione tecnica deve consentire di valutare la conformità del prodotto ai requisiti della direttiva. Essa comprende in particolare i documenti seguenti:

▼*M5*

— una descrizione generale del prodotto, comprese le varianti previste e gli usi cui è destinato;

▼*B*

— gli schemi di progettazione e i metodi di fabbricazione, gli schemi delle parti, dei pezzi, dei circuiti, ecc.;

— la descrizione e le spiegazioni necessarie per la comprensione degli schemi summenzionati e del funzionamento del prodotto;

— i risultati dell'analisi dei rischi e un elenco delle norme previste all'articolo 5, applicate interamente o in parte, e una descrizione delle soluzioni adottate per soddisfare i requisiti essenziali della direttiva quando non siano state applicate interamente le norme previste all'articolo 5;{...}

▼*B*

— i risultati dei calcoli di progettazione, dei controlli svolti, ecc. Se un dispositivi deve essere collegato con uno o più altri dispositivi per funzionare secondo la destinazione prevista, la conformità del primo dispositivo ai requisiti essenziali deve essere dimostrata in collegamento con almeno uno dei dispositivi ai quali deve essere collegato, che possieda le caratteristiche indicate dal fabbricante;

▼*M5*

— le soluzioni adottate di cui all'allegato I, capo 1, punto 2;

— la valutazione pre-clinica;

— la valutazione clinica di cui all'allegato X;

▼*B*

— l'etichettatura e le istruzioni per l'uso.

▼*M5*

4. Il fabbricante istituisce e aggiorna una procedura sistematica di valutazione dell'esperienza acquisita sui dispositivi nella fase successiva alla produzione, anche sulla base delle disposizioni di cui all'allegato X, e prevede un sistema appropriato per l'applicazione delle misure correttive eventualmente necessarie, tenuto conto della natura e dei rischi relativi al prodotto.

Informa le autorità competenti degli incidenti seguenti, non appena egli ne venga a conoscenza:

▼*B*

i) qualsiasi disfunzione o deterioramento delle caratteristiche e/o delle prestazioni {...}

5. Per i prodotti immessi in commercio in confezione sterile e per i dispositivi appartenenti alla classe I, con funzione di misura, il fabbricante deve attenersi, oltre alle disposizioni del presente allegato, anche ad una delle procedure previste agli ►***M5*** *allegati II, IV, V o VI* ◄*. L'applicazione di tali allegati e l'intervento dell'organismo notificato si limitano agli elementi seguenti:*

{...}

— nel caso di dispositivi con funzione di misura, ai soli aspetti della fabbricazione che riguardano la conformità dei prodotti ai requisiti metrologici.

{...}

1.9 Articolo 20

▼*M5*

Riservatezza

1. Gli Stati membri si adoperano affinché, salve le disposizioni e le pratiche esistenti a livello nazionale in materia di segreto medico, tutte le parti interessate dall'applicazione della presente direttiva garantiscano la riservatezza di tutte le informazioni ottenute nello svolgimento dei loro compiti.

{...}

1.10 ALLEGATO X

Gli elementi contenuti nella VALUTAZIONE CLINICA sono per buona parte di recente introduzione e dal punto di vista del software rappresentano una delle sfide maggiori dal punto di vista tecnico. Le metodologie applicate si possono suddividere in 2 famiglie:

- Uso di un qualche sistema (*testing*)
- Uso di metodi formali (leggi matematiche)

La normativa non indica esplicitamente un metodo ma fa riferimento alle metodologie scientifiche e tecniche genericamente in uso.

▼*M5*

1.1. La conferma del rispetto dei requisiti relativi alle caratteristiche e alle prestazioni specificate ai punti 1 e 3 dell'allegato I in condizioni normali di utilizzazione del dispositivo, nonché la valutazione degli effetti collaterali e dell'accettabilità del rapporto rischi/benefici di cui al punto 6 dell'allegato I devono basarsi, in linea di principio, su dati clinici. La valutazione di tali dati, di seguito denominata «valutazione clinica», che tiene conto — ove necessario — delle eventuali norme armonizzate pertinenti, deve seguire una procedura definita e metodologicamente valida fondata alternativamente su:

1.1.1. un'analisi critica della letteratura scientifica pertinente attualmente disponibile sui temi della sicurezza, delle prestazioni, delle caratteristiche di progettazione e della destinazione d'uso del dispositivo qualora:

— sia dimostrata l'equivalenza tra il dispositivo in esame e il dispositivo cui si riferiscono i dati, e

— i dati dimostrino adeguatamente la conformità ai requisiti essenziali pertinenti;

1.1.2. un'analisi critica di tutte le indagini cliniche condotte;

1.1.3. un'analisi critica dei dati clinici combinati di cui ai punti 1.1.1 e 1.1.2 {...}

1.1 **ter**. *La valutazione clinica e il relativo esito sono documentati. La documentazione tecnica del dispositivo contiene tali documenti e/o i relativi riferimenti completi.*

1.1 **quater**. *La valutazione clinica e la relativa documentazione sono attivamente aggiornati con dati derivanti dalla sorveglianza post-vendita.*

Ove non si consideri necessario il follow-up clinico post-vendita nell'ambito del piano di sorveglianza post-vendita applicato al dispositivo, tale conclusione va debitamente giustificata e documentata.

1.1 **quinquies**. *Qualora non si ritenga opportuna la dimostrazione della conformità ai requisiti essenziali in base ai dati clinici, occorre fornire un'idonea giustificazione di tale esclusione in base ai risultati della gestione del rischio, tenendo conto anche della specificità dell'interazione tra il dispositivo e il corpo, delle prestazioni cliniche attese e delle affermazioni del fabbricante. Va debitamente provata l'adeguatezza della dimostrazione della conformità ai requisiti essenziali che si fondi solo sulla valutazione delle prestazioni, sulle prove al banco e sulla valutazione preclinica.*

▼*B*

1.2. Tutti i dati devono rimanere riservati, conformemente al disposto dell'articolo 20.

{...}

2.3. Metodi

2.3.1. Le indagini cliniche debbono svolgersi secondo un opportuno piano di prova corrispondente allo stato delle conoscenze scientifiche e tecniche e definito in modo tale da confermare o respingere le affermazioni del fabbricante riguardanti il dispositivo; dette indagini comprendono un numero di osservazioni sufficienti per garantire la validità scientifica delle conclusioni.

2.3.2. Le procedure utilizzate per realizzare le indagini sono adeguate al dispositivo all'esame.

2.3.3. Le indagini cliniche sono svolte in condizioni simili alle condizioni normali di utilizzazione del dispositivo.

2.3.4. Devono essere esaminate tutte le caratteristiche pertinenti comprese quelle riguardanti la sicurezza, le prestazioni del dispositivo e gli effetti sul paziente.{...}

2 UN CASO DI STUDIO

Come caso di studio ci occuperemo di un software ODOUS che si rivolge ai medici dentisti, sviluppato da BioComputing Competence Centre, la divisione medicale di SCS srl, una azienda con sede in provincia di Bologna.

2.1 Caratteristiche del prodotto

ODOUS è un software di pianificazione chirurgica in 3D per implantologia basato sulle immagini del paziente.

L'applicazione si compone di viste, ogni vista fornisce una ben definita rappresentazione dei dati, e di operazioni, che modificano i dati.

L'applicazione si compone di un menu principale e di un pannello laterale attraverso il quale settare le caratteristiche della vista e delle operazioni.

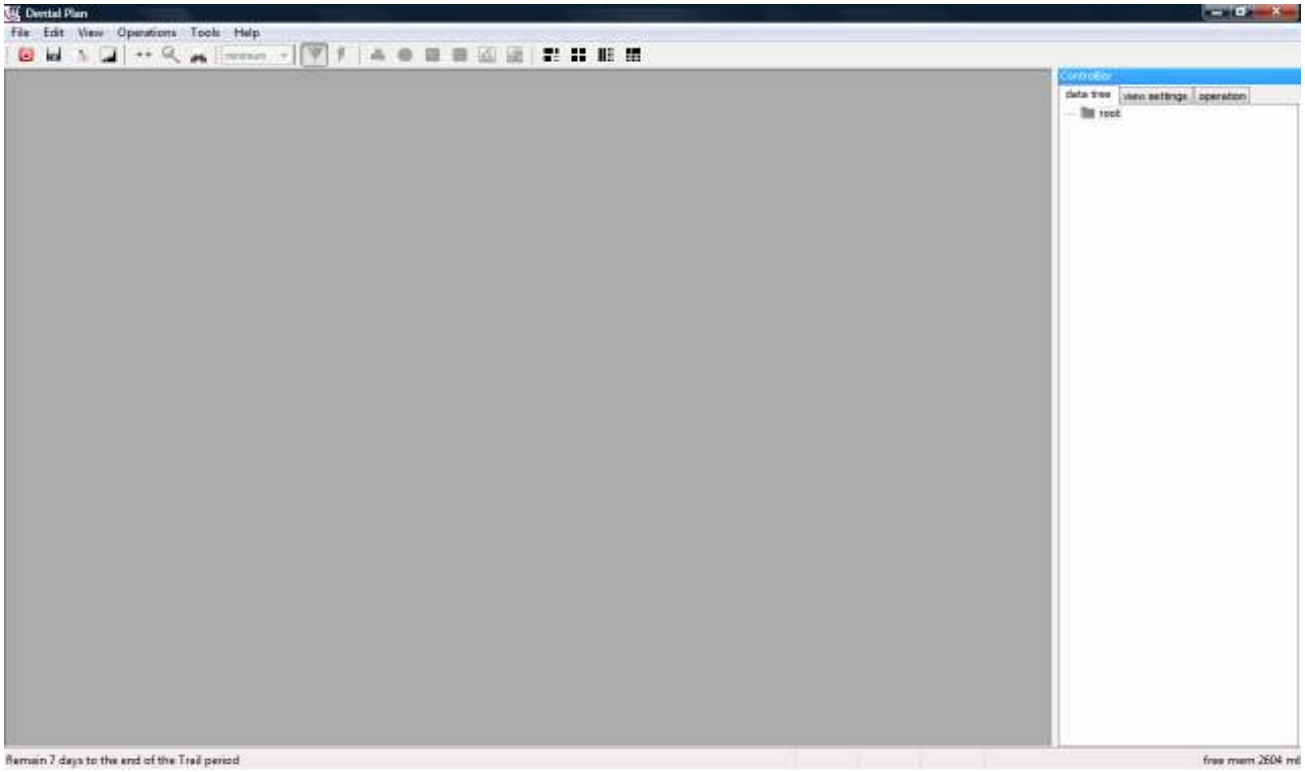


Figure 1 Main application user interface

2.2 Importare i dati

Prima di tutto occorre importare in ODOUS le immagini in formato DICOM, lo standard per le applicazioni medicali, selezionare la parte di interesse e terminare l'importazione.

Una volta importato si visualizza attraverso le viste messe a disposizione sulla barra degli strumenti (I dati visualizzati negli esempi sono ottenuti dall'elaborazione di ODOUS di dataset disponibili su internet).

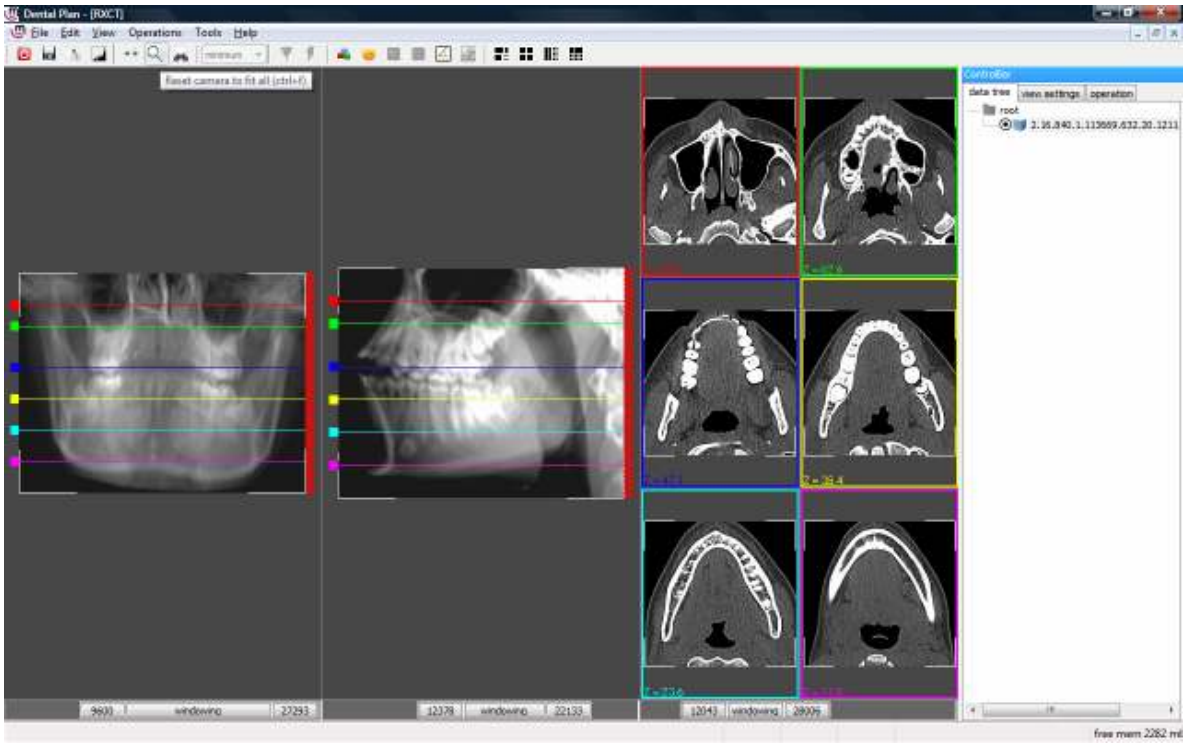


Figure 2 MPR View

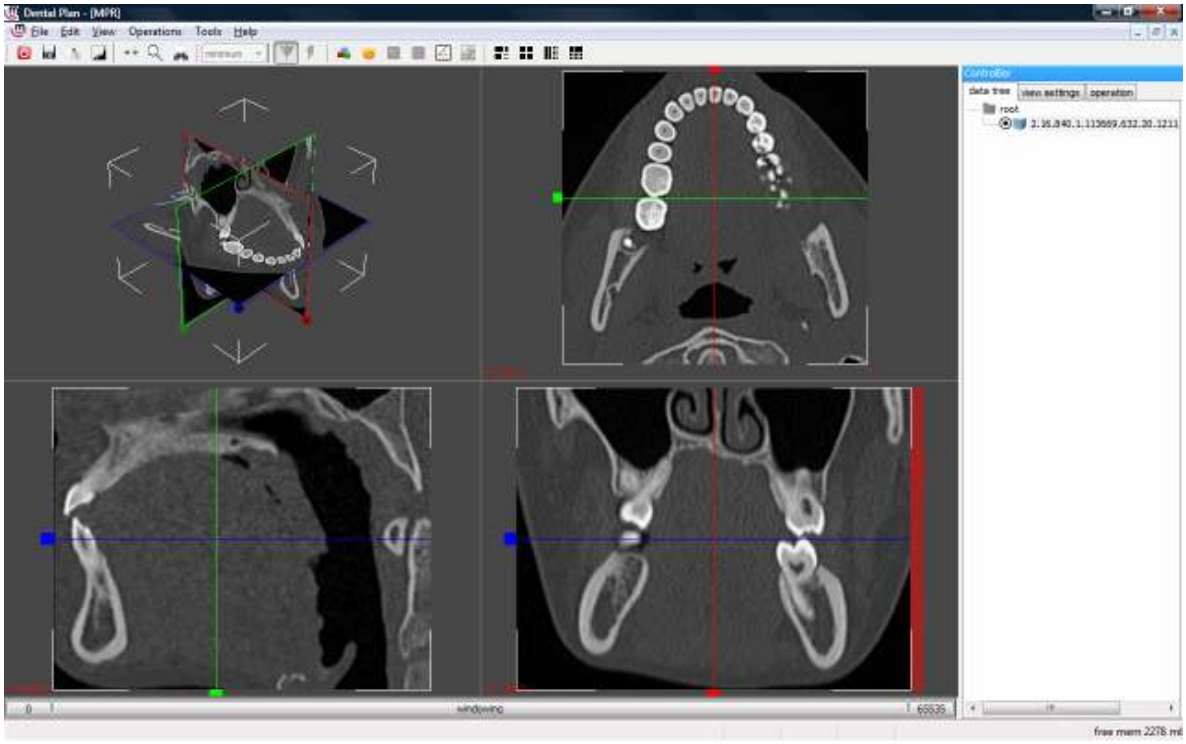


Figure 3 RXCT View

2.3 Cos'è DICOM

Lo standard DICOM (Digital Imaging and COmmunications in Medicine) definisce i criteri per la comunicazione, la visualizzazione, l'archiviazione e la stampa di informazioni di tipo biomedico quali ad esempio immagini radiologiche.

Lo standard DICOM è pubblico, nel senso che la sua definizione è accessibile a tutti. La sua diffusione si rivela estremamente vantaggiosa perché consente di avere una solida base di interscambio di informazioni tra apparecchiature di diversi produttori, server e PC, specifica per l'ambito biomedico.

Lo standard DICOM si basa su un modello concettuale definito "Service Object Pair" (SOP). Il modello SOP è ispirato al paradigma "Object Oriented" (OO) dell'analisi e della programmazione in informatica.

Occorre notare che il DICOM è uno standard industriale, e non uno standard ISO, quindi universale: ciò comporta una certa tolleranza nell'implementazione delle specifiche. Nella maggior parte dei casi, infatti, un'apparecchiatura risulta conforme ad una parte dello standard (ad esempio la modalità di archiviazione delle immagini), mentre adotta tecnologie proprietarie per altre funzionalità (ad esempio la gestione delle liste pazienti).

La compatibilità DICOM ed in generale di qualsiasi dispositivo DICOM compatibile deve essere certificata dal costruttore attraverso un documento autocertificativo, denominato Conformance Statement, che ne elenchi le funzionalità.

I dati radiologici rappresentabili come immagini o le immagini vere e proprie che vengono archiviate secondo lo standard DICOM sotto forma di file vengono comunemente chiamate immagini DICOM. L'errore più comune che viene fatto nell'interpretazione del termine è che queste siano assimilabili ad altri formati di compressione dell'immagine (es. JPEG, GIF, etc.). In verità lo standard DICOM applicato alla codifica dei file non è nient'altro che un metodo per incapsulare i dati e per definire come questi debbano essere codificati o interpretati, ma non definisce alcun nuovo algoritmo di compressione. La maggior parte delle volte, l'immagine viene archiviata in forma non compressa, secondo la codifica con la quale viene prodotta.
(Da Wikipedia, l'enciclopedia libera)

2.4 Usare ODOUS

Per visualizzare i dati si utilizza la ‘panoramica’, che può essere anche ruotata.

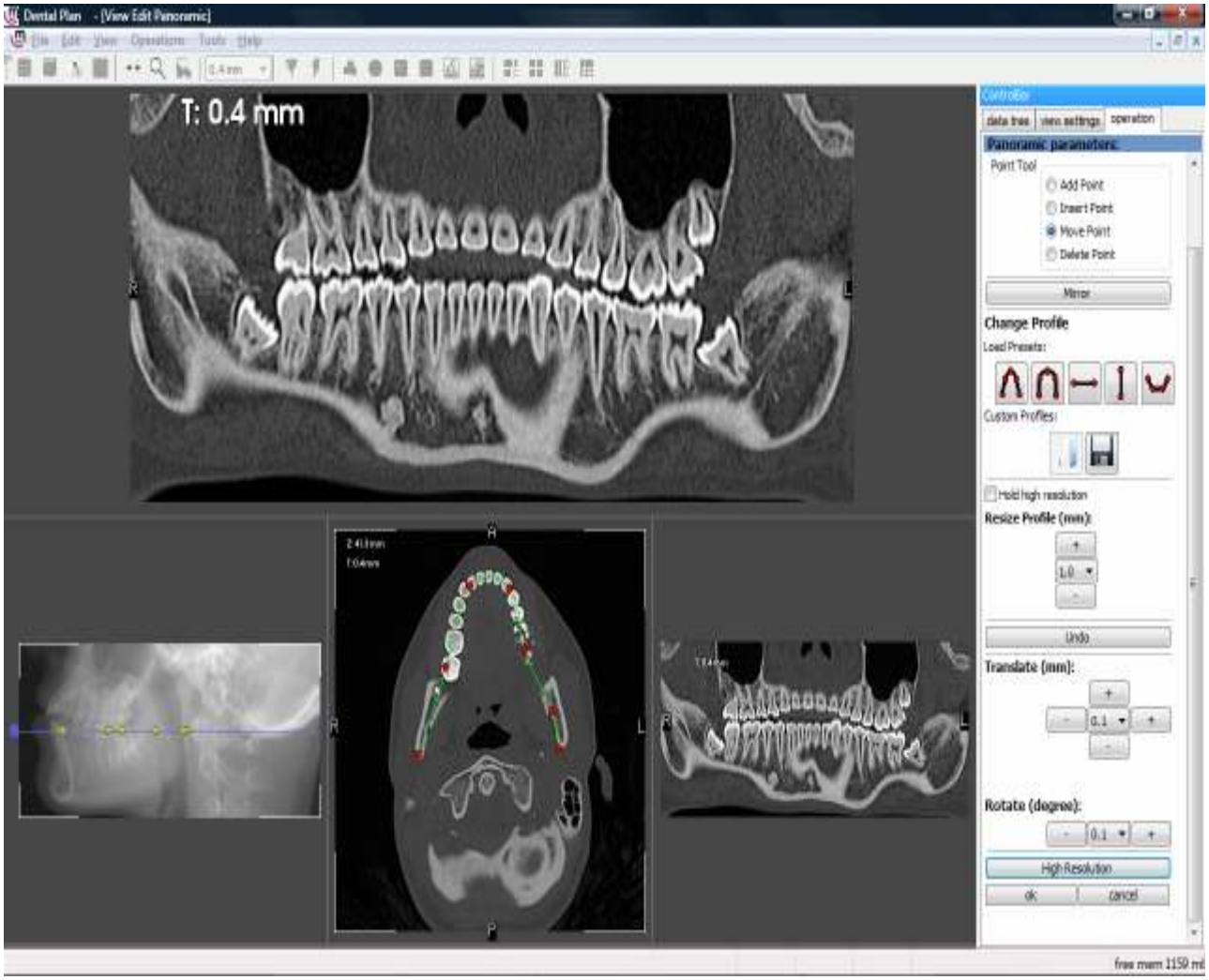


Figure 4 The Pano Operation layout: the upper panel the Pano is rendered at full resolution, in the bottom panels, from the right: a lateral Rx, an axial slice view and a low resolution interactive Pano preview

Dalla panoramica si passa alla definizione del canale mandibolare.

Si possono effettuare diversi tipi di misurazioni, non solo distanze fra punti ma anche distanze angolari.

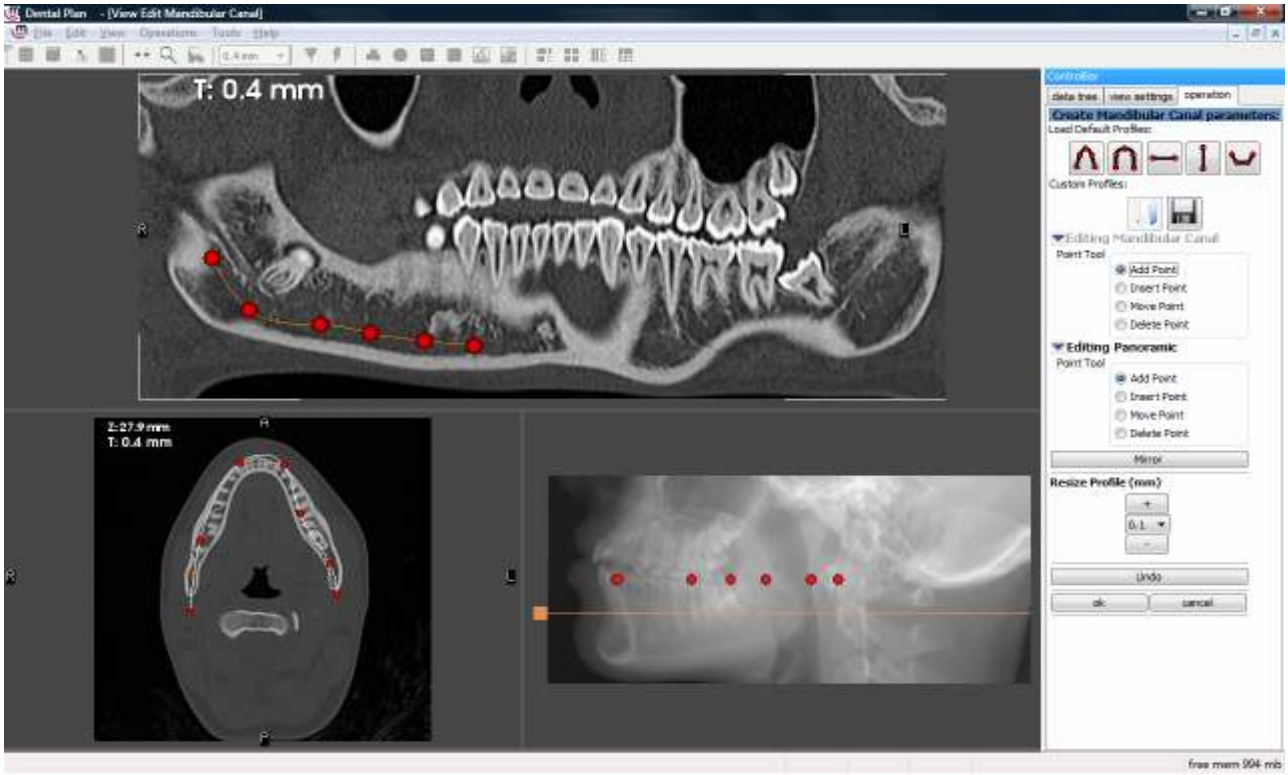


Figure 5 Mandibular Canal Operation: in the upper panel a Pano is used to pick the mandibular panel

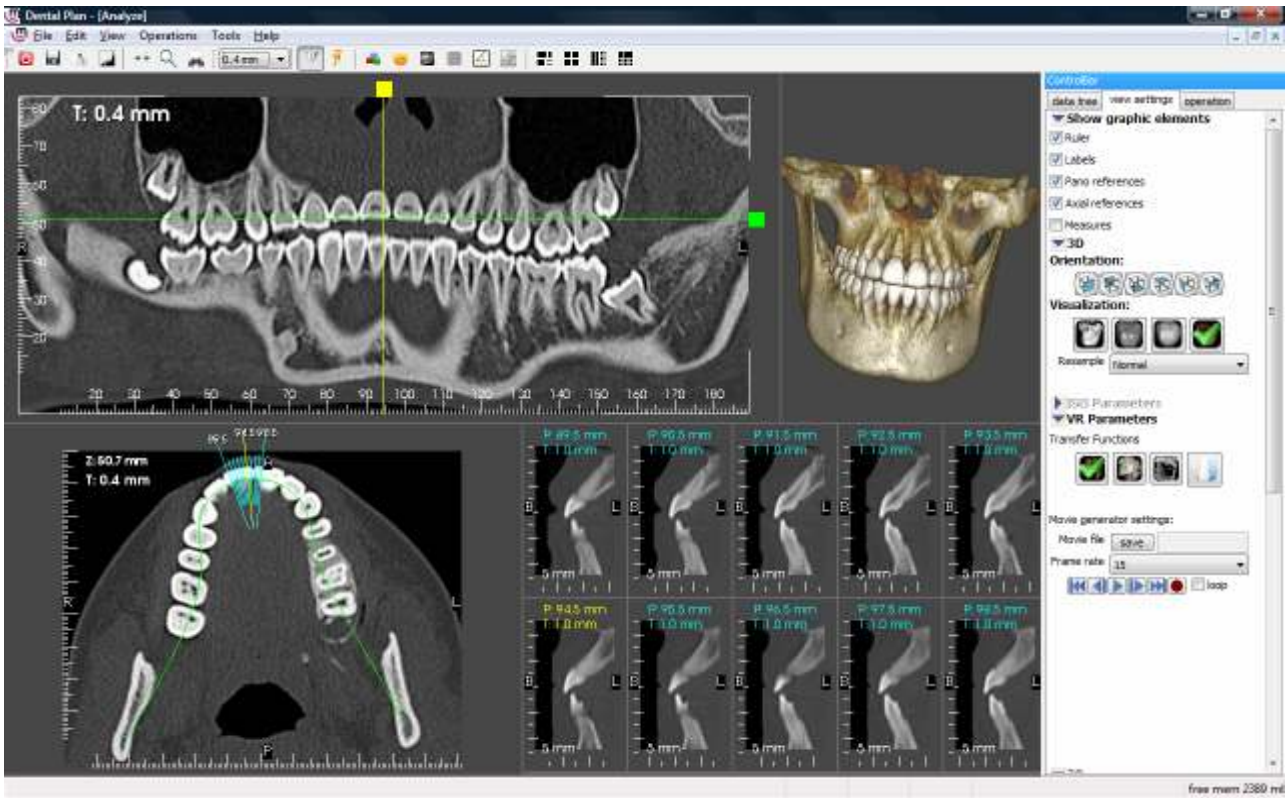


Figure 6 Measurement on the Analyse View

3 LA CERTIFICAZIONE

3.1 La qualità del software

Prima di addentrarci nella disamina della normativa vediamo cosa significa "qualità" in ambito informatico.

Per qualità del software si intende la misura in cui un prodotto software soddisfa un certo numero di aspettative rispetto sia al suo funzionamento sia alla sua struttura interna. (da Wikipedia)

Tradizionalmente i parametri di qualità del software si distinguono fra *interni*, percepiti da chi lo sviluppa, ed *esterni*, quelli percepiti dall'utente e che si riferiscono al software quando è in uso.

Parametri di qualità esterni:

- Correttezza, se data una definizione dei requisiti il software li soddisfa;
- Affidabilità, se i risultati calcolati, le elaborazioni effettuate, le azioni eventualmente eseguite producono gli effetti voluti o comunque con scostamenti tollerabili;
- Robustezza, se si comporta in maniera accettabile anche in corrispondenza di situazioni non specificate nei requisiti;
- Usabilità, il software deve essere corredato di una interfaccia utente e della documentazione appropriate.

Parametri di qualità interne:

- Verificabilità, intesa come la possibilità di verificare che gli obiettivi proposti siano stati raggiunti;
- Manutenibilità: deve essere possibile modificare il software in modo da soddisfare nuovi requisiti;
- Evolvibilità, la facilità con cui il SW può essere adattato a modifiche delle specifiche;
- Riusabilità, la facilità con cui il SW può essere re-impiegato in applicazioni diverse da quella originaria;
- Portabilità del software, che deriva dal fatto di utilizzare:

- linguaggi compilabili su più piattaforme
- macchine virtuali (html/java).

3.2 Stato dell'arte

Dal punto di vista della normativa internazionale sul tema della qualità del software i principali riferimenti in uso sono i seguenti:

- UNI EN ISO 13485:2004 - Sistemi di gestione della qualità
- UNI EN ISO 14971:2009 - Applicazione della gestione dei rischi ai dispositivi medici
- IEC 62304, Ed. 1: Medical device software - Software life cycle processes
- Linee guida FDA (per il mercato degli Stati Uniti):
  - Guidance for the content of Premarket Submissions for Software contained in Medical Devices: 2005
  - General Principles of Software Validation; Final Guidance for Industry and FDA Staff: 2002

3.3 La direttiva europea

Ritornando a quanto abbiamo descritto nel capitolo 1, in sintesi per ottenere la certificazione si parte innanzi tutto dall'allegato IX, dal quale si deduce che il software stand-alone si può classificare di classe I o IIa. Dal tipo di classificazione, attraverso le disposizioni di cui all'articolo 11, si ricava quali siano i requisiti richiesti per la certificazione. Per le classi I e IIa i requisiti sono specificati negli allegati I, VII con annessi la certificazione di prodotto o di processo e la valutazione clinica dell'allegato X.

Quindi per ottenere la certificazione il Fabbricante deve necessariamente, come per gli altri dispositivi medici:

1. identificare la destinazione d'uso;
2. identificare la classe di appartenenza del dispositivo medico;
3. essere conformi con i requisiti essenziali,
4. realizzare il fascicolo tecnico relativo al Dispositivo medico prodotto;

5. seguire l'iter di approvazione di prodotto e/o dell'azienda da parte di un Organismo Notificato (ad eccezione dei prodotti in classe I);
6. redigere la dichiarazione CE di conformità;
7. riportano la marcatura CE seguita dal numero di identificazione dell' Organismo Notificato (ad eccezione dei prodotti in classe I);
8. accompagnare il prodotto con le istruzioni per l'installazione, l'uso e la manutenzione;
9. quando applicabili delle altre regole obbligatorie o volontarie alle quali esso è sottoposto.

Vediamo come si possono sviluppare i punti di cui sopra in riferimento al software di riferimento ODOUS, limitandoci ai soli aspetti strettamente di interesse informatico.

3.4 La destinazione d'uso

Il software ODOUS serve per la modellizzazione 3D per implantologia basato sulle immagini del paziente.

3.5 La classificazione

La destinazione d'uso e la classificazione del dispositivo medico sono responsabilità legale del fabbricante, che deve sviluppare le competenze interne per scegliere in maniera conforme la classe di appartenenza del dispositivo, al fine di riconoscere quali obblighi soddisfare.

Riteniamo di proporre come classificazione per ODOUS la classe I. Notiamo che dal momento che questo software è orientato alla misurazione del canale mandibolare, se volessimo trattarlo come uno "strumento di misura" sarebbe oggetto di una particolare procedura, come spiegato nell'allegato VII.

3.6 Il fascicolo tecnico

Per soddisfare la Direttiva il fabbricante deve tenere a disposizione degli Organi di Vigilanza una completa documentazione tecnica relativa all'azienda ed ai prodotti fabbricati. Questa documentazione viene chiamata "Fascicolo

tecnico". Ad esempio, nel seguito si propone "un indice" di un possibile documento utilizzato per preparare il "Fascicolo tecnico" di una azienda che produce dispositivi medici di classe IIa:

- organizzazione del documento
- generalità sull'azienda
- localizzazione delle attività, lavorazioni esterne
- classificazione dei dispositivi
- ciclo produttivo
- informazioni all'utilizzatore
- norme applicabili
- requisiti essenziali e analisi dei rischi
- glossario aziendale
- dichiarazione di conformità
- manuale della qualità

In questa fase ci interessa documentare come è stato sviluppato il software da certificare. I principali strumenti utilizzati per lo sviluppo di ODOUS sono:

- C++ come linguaggio di programmazione
- XP come la metodologia di sviluppo del software
- utilizzo di CVS per la gestione delle versioni (ma in prospettiva si passerà a SVN)
- PARABUILD (Viewtier System, Inc. CA, USA) per il *continuous build*
- Bugzilla come tool per la gestione dei *bug*
- Test automatizzati a livello di framework e di applicazione
- AQ Time (Automated QA, MA, USA)

3.6.1 eXtreme programming (XP)

Vediamo in dettaglio cosa significa programmare seguendo questo metodo. XP, nato ufficialmente cinque anni fa in un progetto sviluppato da Kent Beck in *Daimler Chrysler*, dopo aver lavorato per parecchi anni con Ward Cunningham, per molte persone non è nulla di più del "buon senso comune".

Anche se XP è incentrato sul codice, è anche e soprattutto un metodo di project management, a dispetto delle critiche sollevate da molte persone. Il punto è che, assunto che lo sviluppo del software è un processo caotico, XP non tenta di trovare un determinismo inesistente ma piuttosto fornisce i mezzi necessari per trattare questa complessità, e la accetta, senza cercare di forzarla in vincoli di pesanti e burocratici metodi.

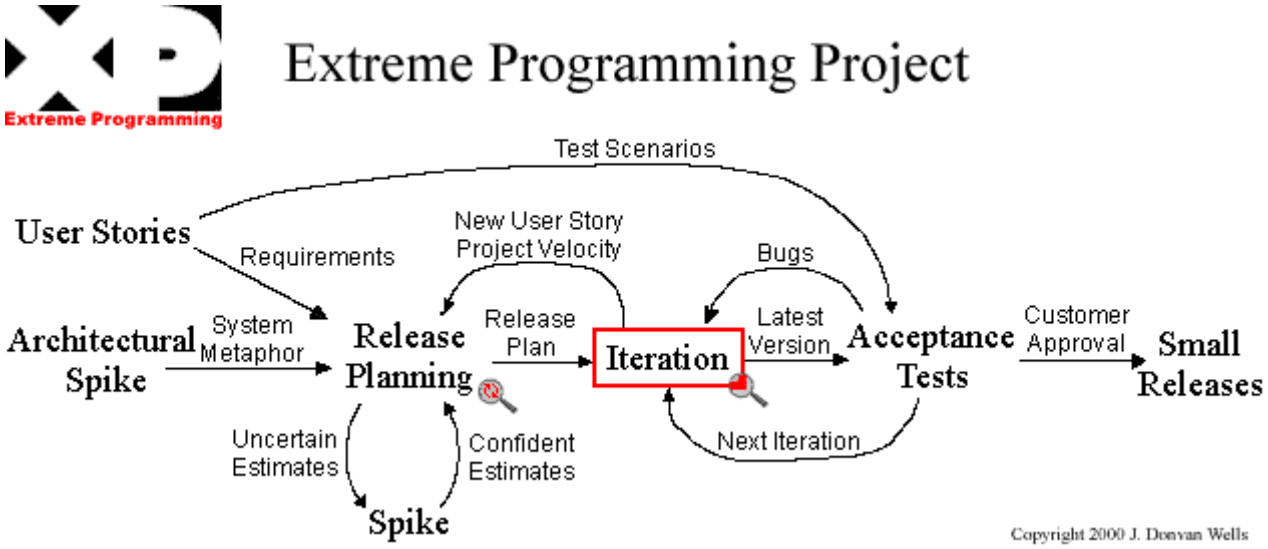
Le tecniche più importanti che lo caratterizzano dagli altri metodi di sviluppo sono:

- Il codice viene costantemente rivisto, tramite il **pair programming**
- I test sono fatti in ogni momento
- (Ri)progettazione ogni volta che lo si ritiene necessario (**refactoring)**
- Le iterazioni sono radicalmente più corte che negli altri metodi, in questo modo si può beneficiare dei commenti di ritorno il più spesso possibile.

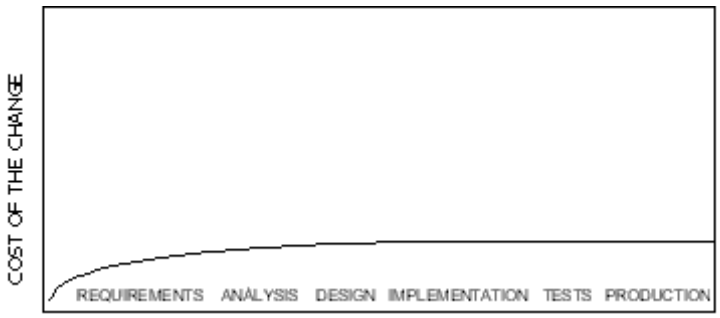
XP precisa quattro variabili per un qualunque progetto: *costo, tempo, qualità* e *portata*. Inoltre specifica che di queste quattro variabili, solo tre possono essere stabilite da elementi al di fuori del progetto (clienti e project manager), mentre il valore della variabile libera verrà stabilito dal gruppo di sviluppo in accordo con gli altri tre valori.

L'XP inoltre presuppone che se il gruppo di progetto si abitua a fare test intensivi (la pietra miliare di XP) e gli standard di codifica vengono seguiti, gradualmente il progetto inizierà a progredire più velocemente di quanto facesse prima.

Un'altra delle idee centrali di XP è che se, come è stato dimostrato, i lunghi cicli di sviluppo dei metodi tradizionali non sono in grado di far fronte al cambiamento, forse ciò che si dovrebbe fare è rendere i cicli di sviluppo più corti., come si vede allo schema del modello XP:



Inoltre se è sempre stato considerato come verità universale che il costo del cambiamento nello sviluppo di un progetto cresca esponenzialmente nel tempo, XP dichiara come questa curva non sia più valida, e che con una combinazione di buone pratiche di programmazione e di tecnologia sia possibile invertirla, come mostrato in figura.



Naturalmente non tutti sono d'accordo con questa supposizione, ma in ogni caso è chiaro che se si decide di usare XP come processo di sviluppo software occorre accettare questa curva come valida.

L'idea di base è che, invece di cambiare a causa dei cambiamenti, si dovrebbe progettare nel modo più semplice possibile, facendo solo ciò che è assolutamente necessario in un certo momento. Grazie all'estrema semplicità del codice, insieme alla conoscenza del refactoring e, soprattutto, il test e l'integrazione continua, i cambiamenti potranno essere portati avanti ogni volta che sarà necessario.

Andiamo nel dettaglio ed esaminiamo le pratiche che compongono XP.

3.6.1.1 Pianificazione

XP considera la pianificazione come un dialogo permanente tra la parte commerciale e tecnica coinvolte nel progetto, in cui la prima decide la portata (cosa è realmente essenziale per il progetto), la priorità (cosa deve essere fatto prima), la composizione dei rilasci (cosa dovrebbe essere incluso in ognuno) e le scadenze per ciascun rilascio. I tecnici, per la loro parte, sono responsabili della stima del tempo necessario per realizzare le funzionalità che il cliente richiede, della messa in evidenza delle conseguenze delle decisioni prese, dell'organizzazione del lavoro e, infine, della pianificazione di dettaglio di ciascun rilascio.

3.6.1.2 Rilasci piccoli

Il sistema va in produzione inizialmente al più pochi mesi prima che sia completamente finito. I successivi rilasci saranno più frequenti, a intervalli di frequenza tra un giorno e un mese, il cliente e il gruppo di sviluppo beneficeranno dei ritorni prodotti da un sistema in produzione e questo si rifletterà sui successivi rilasci.

3.6.1.3 Progettazione semplice

XP indica, in ogni momento, di progettare per le necessità del presente.

3.6.1.4 Testing

Qualunque caratteristica di un programma per la quale non c'è un test automatico semplicemente non esiste. Questa è indubbiamente la pietra miliare su cui è costruito XP: se non stiamo facendo test, non stiamo utilizzando XP. Dovremmo utilizzare qualche framework per testing automatico per fare ciò. Torneremo nel capitolo seguente ad analizzare meglio questo aspetto.

3.6.1.5 Refactoring

Questa pratica di fatto consiste nel lasciare il codice esistente nel più semplice stato possibile, in modo che non sia persa nessuna funzionalità, o guadagnata, e tutti i test continuino ad essere effettuati correttamente. Questo renderà più confidenti nel codice già scritto e quindi meno riluttanti nel

modificarlo quando alcune caratteristiche dovranno essere aggiunte o modificate.

3.6.1.6 Programmazione in coppia (Pair programming)

Tutto il codice sarà sviluppato in coppia: due persone che condividono un solo monitor e una sola tastiera. La persona che scrive il codice penserà al miglior modo per realizzare un particolare metodo, mentre il suo collega farà lo stesso, ma da un punto di vista più strategico. Naturalmente i ruoli sono interscambiabili. Allo stesso modo la composizione delle coppie potrebbe cambiare.

3.6.1.7 Proprietà collettiva del codice

Chiunque può modificare qualunque parte del codice, in un momento qualsiasi. In realtà, chiunque veda un'opportunità per semplificare, facendo refactoring, una qualunque classe o metodo, indipendentemente dal fatto che l'abbia creata oppure no, non dovrebbe esitare a farlo. Questo non è un problema in XP, grazie all'uso di standard di codifica e all'assicurazione che il test ci dà in merito al fatto che tutto continuerà a funzionare dopo ogni modifica.

3.6.1.8 Integrazione continua

Ogni poche ore – o verso la fine di ogni giorno di programmazione – il sistema completo è integrato.

Per questo scopo c'è ciò che è noto come macchina di integrazione, alla quale andrà ogni coppia di programmatori quando essi avranno una classe che ha superato il test di unità. Se dopo aver aggiunto la nuova classe insieme ai suoi test di unità il sistema completo continuerà a funzionare, ovvero supererà tutti i test, i programmatori considereranno la loro attività completata. Altrimenti essi saranno responsabili per riportare indietro il sistema in uno stato in cui tutto funzionava al 100%. Se dopo un certo periodo di tempo non saranno in grado di scoprire cosa non va, dovranno gettare via il loro codice e ricominciare da capo.

3.6.1.9 40 ore alla settimana

Se veramente si vuole offrire qualità, e non semplicemente un sistema che funziona, una cosa è certa: nessuno è capace di produrre lavoro di qualità in 60 ore alla settimana.

3.6.1.10 Cliente sul posto

Un'altra regola controversa di XP è che almeno un cliente reale dovrebbe essere permanentemente disponibile al gruppo di progetto per rispondere a qualunque domanda i programmatori possano avere per lui e per stabilire le priorità.

3.6.1.11 Standard di codifica

Essi sono essenziali per il successo delle proprietà collettiva del codice. Questo sarebbe impensabile senza una codifica basata su standard che permettano a chiunque di sentirsi a proprio agio con il codice scritto da altri membri del gruppo.

3.6.1.12 Le critiche

XP ha sicuramente causato un gran furore all'interno della comunità di ingegneria del software e ha sollevato molte critiche, concentrate principalmente su questi aspetti:

- *La programmazione in coppia,* criticata soprattutto da parte dei project manager, anche se è un'opinione che è senza dubbio condivisa da molti programmatori che hanno un senso eccessivo di proprietà sul codice sviluppato
- il mito delle 40 ore alla settimana
- le tecniche proposte richiedono tempo, una volta che il prodotto è stato realizzato. Ciò rallenta la sua commercializzazione
- le tecniche sono costose da realizzare, richiedono organizzazione, management, specializzazione
- alcuni sostengono che XP funziona solo per buoni sviluppatori, che sono in grado di progettare bene, in modo semplice, e allo stesso

tempo e probabilmente proprio per questo motivo facilmente estensibile, fin dall'inizio

- test, metodi orientati a standard di qualità e metodi stremi non garantiscono la qualità totale

Tuttavia, vale la pena di ricordare almeno che nessuna delle pratiche raccomandate da XP sono un'invenzione del metodo; esistevano tutte da prima, e ciò che ha fatto XP è stato di metterle insieme e di provare che funzionano.

3.6.1.13 Cosa scegliere?

I motivi della scelta della metodologia di sviluppo XP per ODOUS sono da ricercare nelle origini di MAF: infatti MAF è stato sviluppato con questo metodo da un team proveniente da CINECA e Istituti Ortopedici Rizzoli nell'ambito del progetto Multimod. La contiguità di "clienti" (IOR) e "sviluppatori" (CINECA) nello stesso team, il fatto di cominciare un progetto nuovo, la possibilità di avere una maggiore flessibilità nelle specifiche, dato l'alto tasso di innovazione del progetto, hanno reso la scelta di usare XP appropriata. Dopo aver adottato XP per un numero di anni, e comunque rimanendo persistente la necessità di avere rilasci frequenti ed incrementali del software perché potesse essere testato dai "clienti" finali, anche per ODOUS è stata adottata la stessa metodologia.

3.6.2 Il framework

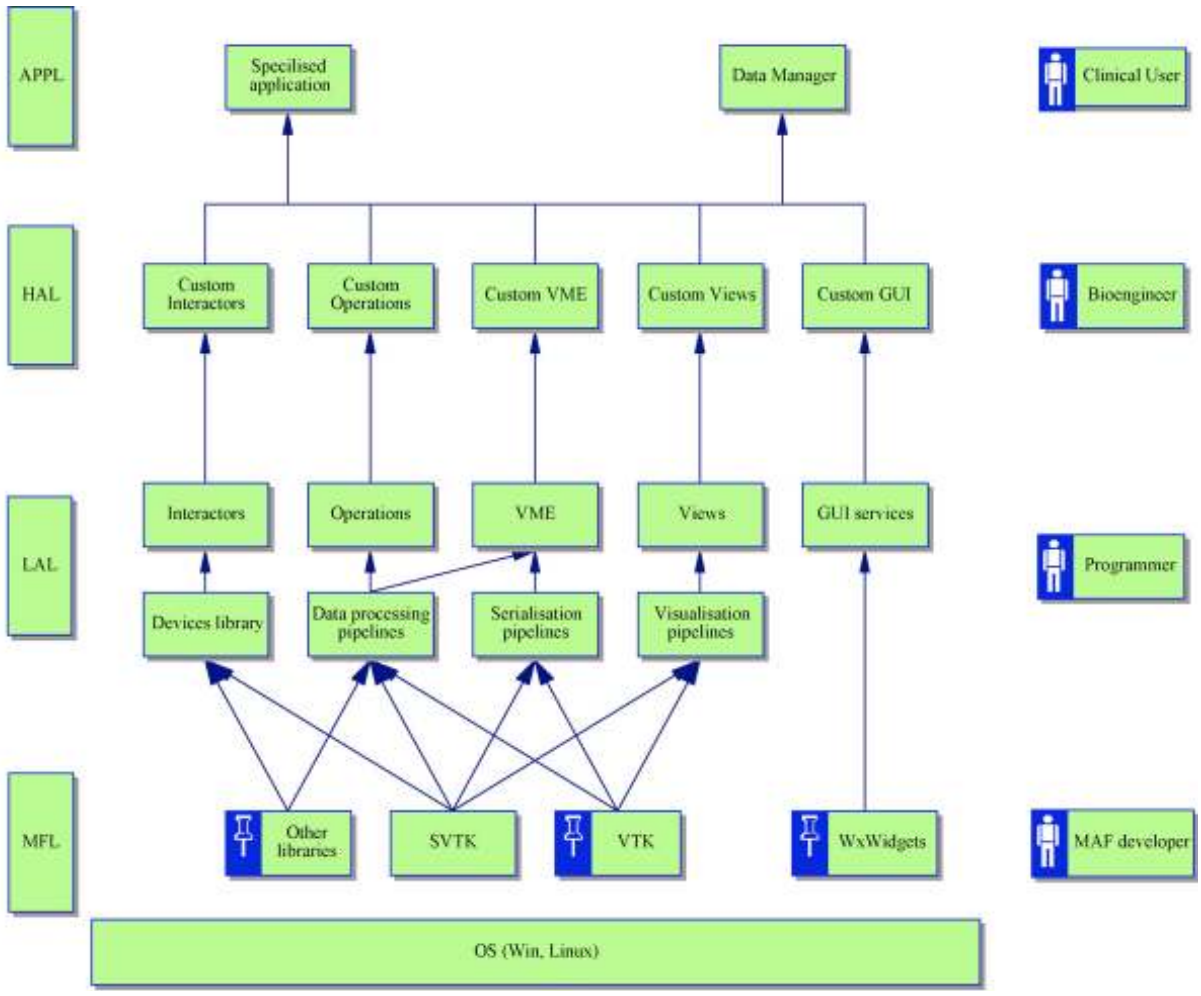
Per lo sviluppo di ODOUS viene usato il framework MAF (Multimod Application Framework). Sviluppato da un gruppo di ricercatori che lavoravano sul progetto Multimod, finanziato dalla Commissione Europea nell'ambito del 5^ Programma Quadro, MAF è un framework open source per informatica applicata alla medicina. Si tratta di una applicazione disegnata per sviluppare velocemente applicazioni multi-modo in ambito medicale. La applicazioni MAF sono multimodali nel senso che all'interno di una applicazione il dato ha diverse rappresentazioni. Il concetto di multimodalità è mutuato dal *medical imaging*, dove il termine "multimodalità" è associato ad una particolare metodica diagnostica. In maniera analoga, in dato per essere "multimodale"

deve poter essere rappresentato con paradigmi diversi, ma mantenere una coerenza spazio-temporale.

Il framework accetta quasi tutti i tipi di dati biomedicali, incluso DICOM, registrazioni in motion-capture, o simulazioni al computer.

MAF può essere compilato su ogni piattaforma che supporti OpenGL, compreso Windows, MacOS X e ogni distibuzione di Unix/linux.

MAF fornisce un insieme di librerie a tre strati utilizzabili per realizzare un i più disparati sistemi software, ed è organizzato in un architettura a quattro livelli.



Il livello inferiore, denominato *multimod foundation layer* (MFL), è composto da alcune delle migliori librerie *open source* disponibili, assemblate in maniera coerente. Tutti i servizi di visualizzazione si basano sul Visualization Tool Kit (VTK). Alcune librerie specializzate forniscono

collision detection, *encryption*, etc. Tutti gli altri servizi di base forniti da MAF sono contenuti in una libreria chiamata SVTK.

Al livello superiore si trova *low abstraction layer* (LAL), che fornisce gli elementi fondamentali di ogni applicazione MAF: interfaccia utente, viste, operazioni, interazioni, e *virtual medical entities* (VMEs).

Al livello più alto (*High abstraction layer*) MAF fornisce una logica di applicazione completamente sviluppata, racchiusa nella classe Logic. Ogni applicazione MAF è un'istanza di questa classe. Quando si utilizza la classe Logic tutti i VME, le viste e le operazioni vengono caricate e quello che si ottiene, una volta compilata, è una applicazione chiamata *Data Manager* (DM). DM può essere utilizzata per fare i primi test su un elemento specifico di una applicazione MAF che si sta scrivendo. (tratto da [10])

3.6.3 Revisioni e versioni

Attualmente gli strumenti software per il controllo di versione sono sempre più riconosciuti come necessari nella maggior parte dei progetti di sviluppo software. In prima battuta, un software di controllo di revisione è uno strumento atto a tenere traccia le modifiche fatte sul codice, e assegnare delle versioni ai file (sorgenti, documenti e quant'altro). È possibile richiamare una versione precedentemente archiviata, indicando una data o una label simbolica (esempio VERSIONE_1_2). Alcuni dei tool OperSource più diffusi sono CVS, SubVersion, mentre fra quelli commerciali ricordiamo almeno BitKeeper, usato anche per sviluppare il kernel Linux da Torvald.

Un buon software di controllo di revisione deve essere in grado di risolvere i conflitti che nascono quando più sviluppatori modificano lo stesso codice per adattarlo ad esigenze differenti.

Un aspetto importante e non trascurabile di tali strumenti è il supporto di tool visuali. Infatti, mentre un compilatore senza un ambiente visuale può essere ugualmente produttivo, un tool di controllo di revisione che non riesca a esprimere in modo grafico la complessità e le iterazioni tra le varie versioni di

un progetto rende più difficile capire quali azioni intraprendere in caso di conflitti, soprattutto se il numero di sviluppatori è maggiore di due o tre.

Nel progetto ODOUS, in affiancamento al framework, viene utilizzato il tool CVS per la gestione delle release, in cui ognuna viene processata e compilata in automatico, in modo da mantenere consistenza nella versione. In questa fase vengono anche effettuati alcuni test "in automatico".

3.7 Definire e documentare i protocolli di testing

Essendo stata esplicitata la destinazione d'uso, si passa alla fase successiva, che consiste nel fornire evidenza della performance del software e della sua efficacia clinica.

Collaudare un prodotto software significa verificare che il suo comportamento sia aderente alle specifiche di progetto, individuare eventuali difetti e, successivamente, eliminarli. Con la parola *difetto*, si intende genericamente un errore nel sistema. Secondo la terminologia definita dall'IEEE esistono diversi tipi di errori nel software:

- **Error**: è un errore umano nel processo di interpretazione delle specifiche, nel risolvere un problema o nell'usare un metodo
- **Failure**: è un comportamento del software non previsto dalle specifiche
- **Fault**: è un difetto del codice sorgente (detto anche *bug*)

Il collaudo comprende due aspetti ben distinti:

1. verifica che tutti i requisiti siano implementati correttamente secondo le specifiche;
2. identificazione e conseguente eliminazione dei difetti presenti nel codice.

Il primo aspetto consiste nel verificare che il software implementato contenga tutte le funzionalità espresse nelle specifiche e che siano implementate correttamente. In questo caso si verifica il sistema nel suo insieme analizzandone il comportamento (per esempio interazioni tra moduli o sotto-sistemi) ma non scendendo nei dettagli (ad esempio non si testano singole classi o funzioni). Questo tipo di verifica, però, non riguarda solo i requisiti

funzionali di un sistema ma anche quelli non funzionali come prestazioni, affidabilità, sicurezza, usabilità ecc..

Il secondo aspetto si focalizza nella verifica che ogni porzione di codice sia corretta, quindi si analizza in dettaglio ogni singola classe o funzione del sistema.

Come si vedrà successivamente, le tecniche utilizzate per testare questi due aspetti sono diverse, anche se alcune possono essere applicate in entrambi i casi. Si potrebbe ipotizzare, a tal scopo, di disporre di norme armonizzate e di norme definite dallo stato dell'arte. Nell'area dell'ingegneria del software, però, collaudare un sistema presenta problemi totalmente diversi da quelli tradizionali, rendendo inapplicabili tutte le pratiche utilizzate negli altri settori dell'ingegneria. Infatti, collaudare un prodotto software moderno non si limita alla verifica che gli algoritmi implementati siano corretti, ma anche che il sistema in esame interagisca correttamente con altri sistemi software (almeno con il sistema operativo) e hardware (almeno la macchina che lo esegue), i quali possono a loro volta essere difettosi. Inoltre anche nel software esistono dei limiti, ma questi non sono altro che tutti i possibili input del programma (sia input validi che non) e, in generale, nel software ogni valore di ingresso si comporta come un caso a se. Quindi, per avere la certezza che un sistema software sia corretto sarebbe necessario verificare il suo comportamento con tutti i valori possibili di input. Sfortunatamente, un approccio del genere è impraticabile per qualunque sistema software, anche se semplice, a causa del tempo (e quindi costi) che richiederebbe. Ne consegue, data l'impossibilità di verificare in maniera esaustiva un sistema, la probabilità che vi siano errori è sempre maggiore di zero, indipendentemente da quanto si testi il prodotto. I sistemi automatici di test, la disciplina nello scrivere i test e nell'eseguirli, le nuove tecniche di sviluppo del software sono tutti strumenti che hanno, tra gli obiettivi, la riduzione del numero di difetti nei prodotti software. A riprova di questa situazione, la ricerca in questo campo è molto attiva, come dimostrato dalla quantità di articoli su questo tema che vengono regolarmente presentati nelle maggiori conferenze internazionali e sulle riviste più prestigiose.

Da quanto appena esposto risulta evidente quanto il *testing* sia, fra le fasi dello sviluppo del software, la più complessa.

3.7.1 Il testing nel processo di produzione del software secondo XP

Passiamo ora ad analizzare le tecniche di *testing* utilizzate nella produzione di ODOUS. Nei capitoli precedenti abbiamo già citato XP come metodo di sviluppo di ODOUS. L'approccio allo sviluppo software in questo metodo, e più in generale delle Metodologie Agili, è centrato sul *testing* del prodotto durante tutto il ciclo di produzione. Le Metodologie Agili rappresentano l'applicazione dei principi della *lean production* alla produzione del software e, come tali, sono orientate alla produzione di manufatti di alta qualità, tramite la riduzione dello spreco. Nel caso del software esistono costi direttamente imputabili allo spreco e costi da esso generati:

- **costi diretti**: tempo impiegato per la produzione;
- **costi generati**: tempo impiegato per la correzione dei difetti, danni provocati dal sistema difettoso ecc..

Per aumentare la qualità dei prodotti e ridurre lo spreco, le MA non considerano il *testing* come una fase separata dello sviluppo del software, ma la integrano in tutte le fasi dello sviluppo utilizzando diverse tecniche di test (*black-box, white-box*) e facendo uso di strumenti automatici per la loro continua esecuzione. In questo si differenzia sensibilmente per esempio dal modello a cascata (*waterfall*): infatti in questo, ed in altri modelli di sviluppo, il *testing* si trova alla fine del ciclo di produzione o, in ogni caso, si collauda il prodotto dopo aver scritto una parte consistente del codice.

Molte delle pratiche su cui si basa l'*Extreme Programming* hanno, direttamente o indirettamente, una qualche relazione con il *testing*. Tra queste, quelle che hanno una relazione diretta con il testing sono: il *Test Driven Development* (Sviluppo Guidato dai Test), la *Continuous Integration* (Integrazione Continua) e le *Small Releases* (Piccoli Rilasci).

Il *Test Driven Development* (TDD) è un approccio allo sviluppo del software che pone al centro dell'attenzione il test, sia per quanto riguarda lo

sviluppatore, sia per il cliente. Dal punto di vista del programmatore, mettere al centro dell'attenzione il test significa:

- Applicare la tecnica del *Test First*, vale a dire scrivere i test di verifica del codice prima di scrivere il codice stesso.
- Utilizzare i test unitari (*Unit Testing*) per verificare che ogni micro-funzionalità sia corretta
- Sviluppare in modo incrementale, implementando pochissimo codice alla volta e verificandone continuamente la correttezza.

Con la diffusione dell'*Extreme Programming* è nata una discussione attorno ai benefici di questo approccio e alle difficoltà che comporta una sua corretta implementazione. Tra i benefici troviamo i seguenti:

- per scrivere un test si è costretti a pensare alla struttura del sistema prima di scrivere anche una singola riga di codice;
- i test non vengono scritti di fretta alla fine e questo contribuisce ad aumentarne l'efficacia (sia riducendo la probabilità di scrivere test errati che di scrivere test troppo semplici);
- i test possono essere eseguiti non appena il codice è stato scritto, evidenziando immediatamente i problemi.

Scrivere i test prima del codice non è semplice e solo un programmatore esperto è in grado di scrivere test rilevanti prima del prodotto: infatti, è necessario avere una visione ben chiara della struttura del sistema e della specifica funzionalità che si sta per implementare. Il rischio è che programmatori inesperti scrivano dei test che non sono adeguati a verificare in modo efficace il codice che scriveranno. Dopo aver scritto il codice della funzionalità ed aver verificato che tutti i test siano eseguiti correttamente, lo sviluppatore deve riorganizzare il codice in modo da eliminare eventuali duplicati, migliorare l'architettura e la leggibilità del codice ecc.. Questo insieme di modifiche e miglioramenti del codice prende il nome di *Refactoring*.

Quindi, lo sviluppatore implementa il TDD ripetendo continuamente i seguenti passi:

1. scrittura del test unitario e dei **Mock Object** per una funzionalità minimale (i *Mock Object* che non sono altro che classi che si

sostituiscono temporaneamente ai sottosistemi ancora da sviluppare, semplificando la creazione dei test unitari. In questo modo, è possibile scrivere i test prima del codice in qualunque situazione. Insieme ai test, quindi, lo sviluppatore deve scrivere anche tutti i *Mock Object* necessari a sostituire i sottosistemi non ancora disponibili ma necessari per collaudare efficacemente il codice che intende sviluppare.);

2. implementazione della funzionalità;
3. verifica di correttezza tramite l'esecuzione di tutti i test (questo fa si che non si verifichi solo se la funzionalità appena implementata è corretta ma anche che il codice scritto non provochi effetti indesiderati sul codice esistente);
4. *Refactoring*;
5. verifica di correttezza tramite l'esecuzione di tutti i test (questo serve per verificare che il *Refactoring* non abbia introdotto inavvertitamente degli errori).

Applicare il TDD senza degli strumenti adeguati è pressoché impossibile a causa della quantità di tempo che si dovrebbe dedicare al test. Per questo motivo sono nati diversi strumenti di supporto al *testing* che permettono di automatizzare totalmente la loro esecuzione e rendere questo approccio applicabile.

Dal punto di vista del cliente, mettere al centro dell'attenzione il test significa descrivere in dettaglio i criteri che saranno utilizzati per valutare la corretta implementazione di un prodotto e, quindi, accettare il software sviluppato (Test di Accettazione o *Acceptance Test*). I test di accettazione vengono scritti dal cliente (o dai programmatori in stretta collaborazione col cliente), in un linguaggio a lui comprensibile (normalmente si usa il linguaggio naturale) e prima dell'implementazione di ogni singola funzionalità. Gli sviluppatori hanno il compito di tradurre questi test nel linguaggio usato per l'implementazione del prodotto e di eseguirli prima di presentare il sistema al cliente.

Il TDD, quindi, consente di implementare nella produzione del software i meccanismi di controllo che sono tipici della *lean production*, permettendo di individuare tempestivamente i problemi all'interno del prodotto.

La *Continuous Integration* consiste nell'integrare continuamente il codice sviluppato dai singoli programmatori per evidenziare immediatamente i potenziali problemi di integrazione. Questo significa che ogni sviluppatore, non appena completata una funzionalità, deve integrarla nel prodotto ed eseguire i test di integrazione per verificare che il proprio codice si comporti correttamente quando interagisce con quello scritto dagli altri e, in caso vi siano problemi, individuare e correggere immediatamente gli errori.

L'integrazione continua si basa sull'utilizzo di diversi strumenti tra cui un sistema di gestione delle configurazioni (CVS, V*isual Source Safe*, *ClearCase* ecc.) ed un sistema per l'automazione dei test di integrazione. Questa pratica permette di tenere sotto controllo i problemi legati all'interoperabilità del codice scritto da persone diverse per evitare che si individuino problemi di questo tipo quando è troppo tardi per evitare ritardi nella consegna del prodotto o di ridurre la qualità finale.

La pratica delle *Small Releases* consiste nel rilasciare il prodotto al cliente molto frequentemente (ogni 2-4 settimane) e con poche funzionalità aggiunte ogni volta. Questo approccio permette al cliente di valutare il prodotto sviluppato ed identificare tempestivamente eventuali incomprensioni nei requisiti. In questo modo, se il cliente individua qualche problema, gli sviluppatori possono intervenire prima che si sviluppino altre parti di codice riducendo al minimo lo spreco associato.

3.7.2 Gli strumenti di testing automatico

In generale, per ODOUS sono stati applicati un certo numero e tipo di test automatizzati. Sostanzialmente i tipi di test effettuati si possono suddividere in due macro-classi:

- A livello di framework
    - *Code Coverage*
    - *Coding style*

In MAF vengono sistematicamente effettuate verifiche relativamente alle librerie che lo compongono. Viene tenuta documentazione delle percentuali di classi testate fino ad oggi. Sono presenti alcune funzionalità per il monitoraggio di determinate caratteristiche formali delle classi rilasciate, ad esempio il nome della classe. Infatti vi sono più gruppi di lavoro che sviluppano sul framework, e si vuole controllare che vengano rispettati gli standard di sviluppo.

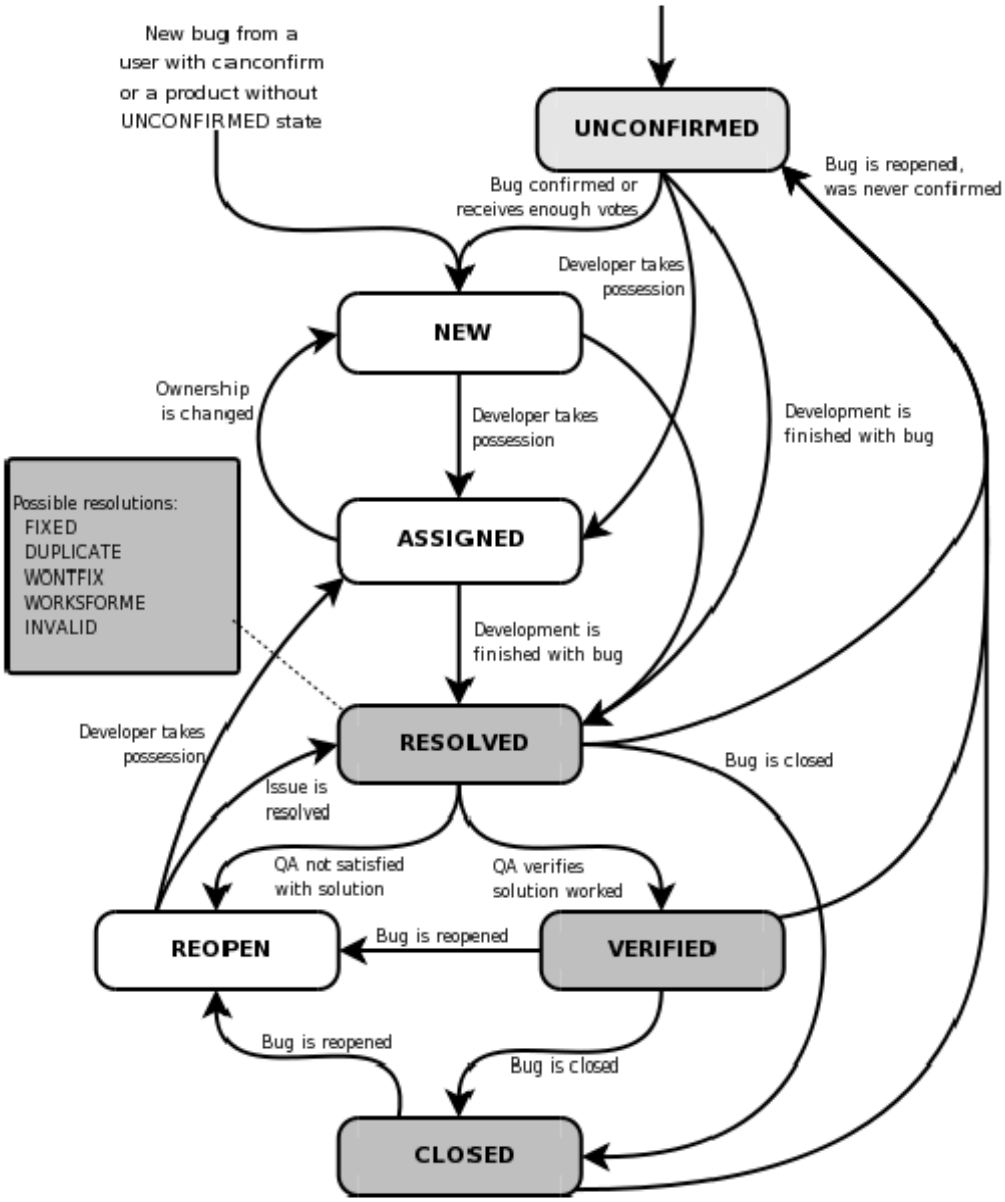
Il software, una volta scritto, viene sottoposto ad una prima fase di test, anche visuale, nel framework stesso.

- A livello di applicazione
    - CPPUNIT
    - Acceptance tests: richiesti dal cliente perché "ai suoi occhi" il sistema soddisfi i requisiti
    - QA dashboard: ossia un tool per gestire e tracciare un prodotto. Fornisce una panoramica dello status dei *white-box* e *black-box* test, i principali *bug*, metriche di performance, ecc. La dashboard viene aggiornata ogni volta che i test vengono eseguiti. A seconda del tipo di *dashboard* (*Continuous* o *Nightly*) viene eseguito il processo di aggiornamento codice da repository, build, test, packaging dell'eseguibile.
      La dashboard inoltre fornisce risposte alle seguenti richieste:
        - stato del *testing*
        - POA dei test per versioni da rilasciare
        - quando si prevede di terminare i test
        - quali test sono già stati completati
    - Memory leaks
    - Automated GUI testing

3.7.3 Strumenti di supporto per il testing

E' piuttosto frequente nello sviluppo di software che ai *tool* di controllo delle revisioni siano affiancati dei software di *bug tracking*, in grado cioè di raccogliere i *bug* trovati dagli utenti, elencarli, ordinarli per priorità eccetera.

Nel progetto ODOUS i *bug* rilevati vengono classificati utilizzando un tool OpenSource Bugzilla, della fondazione Mozilla e rilasciato con Mozilla Public Licence, con possibilità di collegare la versione di software al *bug*



Lifecycle of a Bugzilla Bug (da http://www.bugzilla.org/docs/)

riscontrato, in modo da sapere per ogni versione disponibile quali siano i *bug* conosciuti, e configurato in questo modo:

- ogni applicazione verticale MAF ha la propria lista di rilasci
- ogni servizio fornito via web ha la sua lista di rilasci

- ogni framework ha la propria pagina
- per le applicazioni MAF, la granularità delle componenti è a livello di Operazioni, Viste, livello di VME.

Più in generale gli sviluppatori di ODOUS si sono dotata di un *Workflow* per la gestione dei *bug*. Il primo passo consiste nel verificare se il *bug* segnalato sia effettivamente da considerare tale. Se il *bug* viene accettato, deve essere riprodotto. Per fare questo occorrono informazioni delle versioni di SO e di prodotto, eventualmente anche un set di dati se l'errore è difficile da replicare. Una volta replicato l'errore, lo sviluppatore identifica l'estensione appropriata della *automatic test suite***.** Il test fallito dovrebbe invalidare la dashboard ed essere esposto nel report della *automatic test machine***.** Terminato di implementare la *patch*, lo sviluppatore consolida il codice e verifica che il test sia superato sia nella *automatic test machine* che sulla macchina principale. Passato il test, viene resa disponibile una nuova versione.

Ovviamente sul mercato esistono molti strumenti che forniscono supporto alla fase di test. Il supporto è fornito in modi molto diversi tra cui il tracciamento dei *bug*, l'esecuzione automatica dei test, la verifica di quanto codice è coperto dai test ecc.. Tra gli strumenti dedicati al collaudo, quelli *OpenSourc*e sono sicuramente quelli più noti e diffusi, anche in ambiti commerciali. Fra i più noti possiamo citare, oltre a Bugzilla (http://www.bugzilla.org/), anche

- Scarab (http://scarab.tigris.org/) : si tratta un sistema di tracciamento dei bug che permette di raccogliere le segnalazioni di bug, a prioritarizzarle, ad assegnare la correzione ad un programmatore e, infine, a tenere traccia dell'evoluzione di ogni singolo difetto trovato nel codice.
- Cactus (http://jakarta.apache.org/cactus/) : permette di collaudare applicazioni serverside scritte in Java. Il sistema permette di scrivere vari tipi di test (unitari, di integrazione ecc.) e di automatizzarne l'esecuzione.

- CruiseControl (http://cruisecontrol.sourceforge.net/) : strumento per l'integrazione continua che si integra con i sistemi di gestione delle configurazioni (CVS, Visual Source Safe ecc.), scarica il codice del prodotto in esame, lo compila ed esegue una batteria di test. E'in grado di generare rapporti sullo stato del sistema analizzato e pubblicarli via web o inviarli per e-mail.
- JBlanket (http://csdl.ics.hawaii.edu/Tools/JBlanket/) : strumento che permette di verificare quanto e quale codice sorgente è coperto dai test. Le informazioni ricavate sono molto utili per verificare che tutto il codice sia considerato almeno in un test. Anche in questo caso, il sistema può generare rapporti relativi allo stato del sistema sotto analisi e inviarli agli sviluppatori interessati.
- JUnit (http://www.junit.org/) : fornisce supporto alla creazione e all'esecuzione di Unit Test. Lo strumento aiuta il programmatore ad automatizzare questi test e a verificare che il codice sviluppato sia corretto. Esistono versioni di questo prodotto per moltissimi linguaggi di programmazione tra cui Java, C/C++, C#, Pascal, Pyton ecc..
- Mock Objects (http://www.mockobjects.com/) : sistema per la creazione di mock-object, strutture che semplificano la scrittura dei test unitari sostituendosi temporaneamente alle parti del sistema che devono ancora essere sviluppate.

3.8 Controllo documentale e gestione configurazione

I Requisiti di sistema di ODOUS sono i seguenti:

	Minimum	Recommended
CPU	Intel P4, 3GHz	Intel Centrino Duo2, 2x3GHz
RAM	1GB	3GB
Operating System	WinXP	WinXP Pro SP2
Graphic Card	Embedded	nVidia GeForce 9800 or superior
Screen Resolution	1024x768	1280x1024

In questa sede non ci occuperemo della documentazione che deve accompagnare il prodotto.

3.9 Analisi dei rischi

Un aspetto della normativa piuttosto insolito quando si parla di software in maniera generica è l'analisi dei rischi. Come principio guida la norma stabilisce che spetta al produttore individuare i metodi e le tecniche per soddisfare i requisiti imposti dalla norma.

3.9.1 Stato dell'arte

Dal punto di vista della normativa internazionale il punto di riferimento è sicuramente la norma CEI UNI EN ISO 14971 "Dispositivi medici. Applicazione della gestione dei rischi ai dispositivi medici"

La norma EN ISO 14971 recepita in Italia come UNI CEI EN ISO 14971 sostituisce a tutti gli effetti dall' Aprile 2004 la precedente norma EN 1441 ed impone ai produttori di dispositivi medici di sviluppare un processo di gestione dei rischi che copra tutte le fasi del ciclo di vita dei dispositivi stessi (progettazione, costruzione, consegna, impiego, manutenzione, smaltimento finale). Riconosciuta dalla *US Food and Drug Administration*, dall'Europa, Canada ed Australia questa norma rappresenta oggi di fatto lo standard per il controllo del processo di gestione dei rischi dei dispositivi medici. Essa non si applica ai giudizi clinici relativi all'uso dei dispositivi né precisa i criteri di valutazione ed accettabilità dei rischi. Impone tuttavia alle Aziende costruttrici come parte integrante del sistema di qualità, di sviluppare un piano di gestione dei rischi che:

- Definisca le condizioni di impiego di ciascun dispositivo
- Identifichi e valuti sotto il profilo della gravità e probabilità i rischi associati a tale impiego.
- Controlli tali rischi intervenendo con azioni correttive ove necessario
- Validi l'efficacia di tali controlli.
- Stabilisca responsabilità e frequenze di controllo

La applicazione delle norme UNI CEI EN ISO 14971 rappresenta per le Aziende produttrici di apparati medicali un impegno da non sottovalutare.

È responsabilità delle Direzioni Aziendali creare le condizioni perché il processo di gestione dei rischi degli apparati prodotti, possa svilupparsi, fornendo supporto, decisioni e mezzi. Tale processo diventa parte integrante del sistema qualità aziendale.

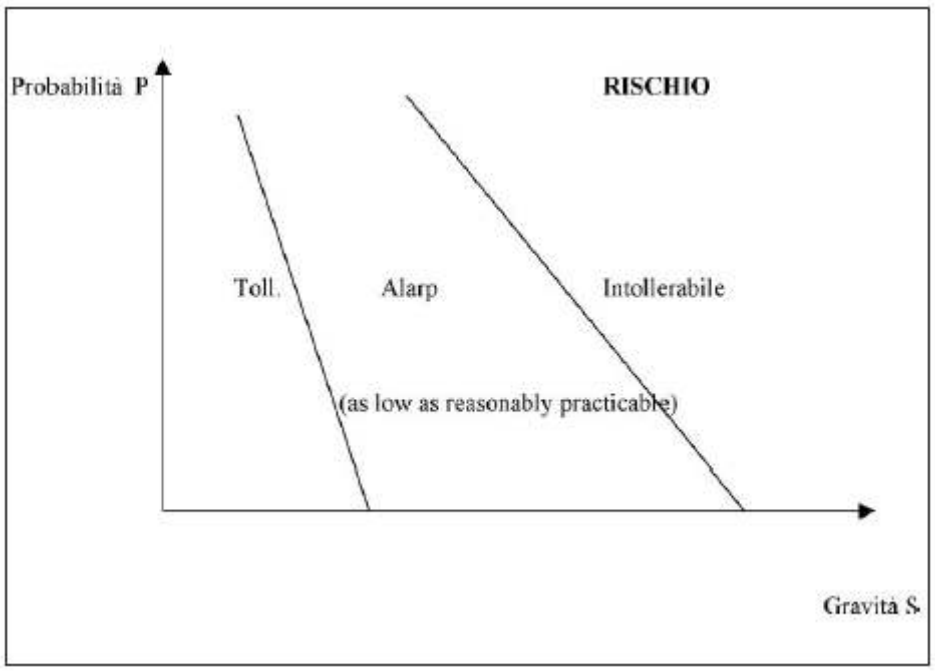
Tra gli impegni più significativi ricordiamo:

- La costituzione ed organizzazione dei team di lavoro
- La definizione dei criteri di valutazione dei rischi (Fattori considerati, scale di riferimento)
- La definizione dei limiti di accettabilità degli stessi
- La definizione dei criteri di calcolo ed accettabilità del rischio residuo
- La scelta dell'ambiente informatico in cui conservare considerazioni e valutazioni riguardanti lo sviluppo del processo
- La organizzazione delle informazioni post-produzione.

Non sempre sarà possibile dimostrare di aver fatto una scelta giusta. Molte infatti sono le incognite e gli imprevisti. Sempre però l'Azienda deve essere nella condizione di poter dimostrare che le decisioni prese sono state prese in modo corretto:

- Nel rispetto degli standard internazionalmente riconosciuti
- Utilizzando i metodi migliori
- Prestando la necessaria attenzione
- Documentando il processo decisionale in ogni sua fase.

L'analisi dei rischi connessi con l'uso dei dispositivi medici ricade sul costruttore, ma anche sui responsabili delle strutture sanitarie (per gli usi difformi da quelli previsti e per la conservazione nel tempo delle prestazioni dei dispositivi). In ogni caso, la CEI UNI EN ISO 14971 aggiunge che l'utente finale deve essere informato dal costruttore dei rischi residui, in modo da poter prendere decisioni informate.



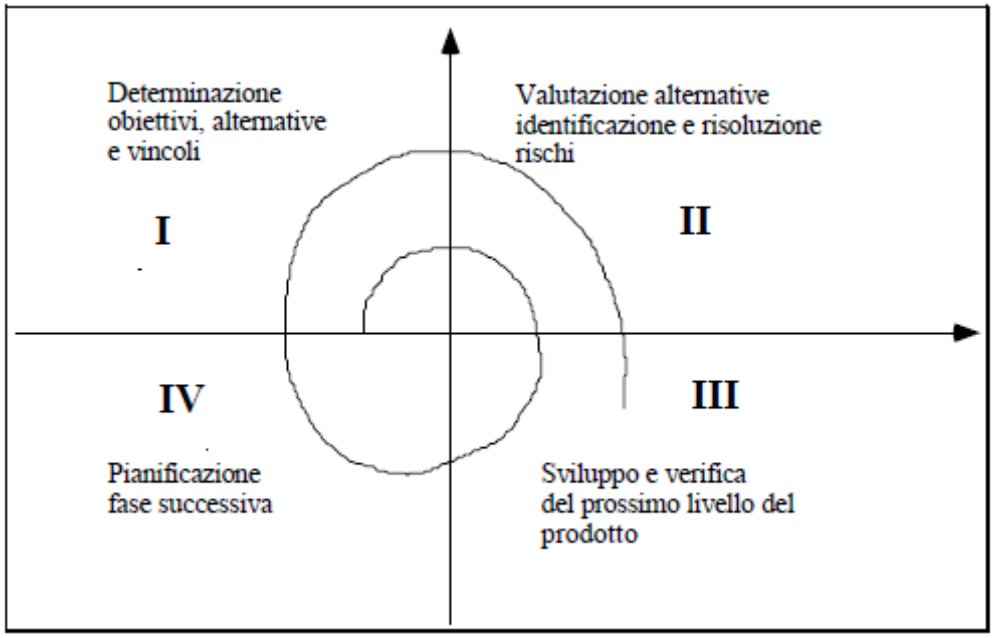
3.9.2 Analisi dei rischi e modelli di sviluppo del software

In ingegneria del software l'analisi dei rischi, intendendo per "Rischio inerente un'attività" una "valutazione (misura) dell'incertezza che si ha sul risultato finale dell'attività", è strettamente collegata al modello di sviluppo del software. Di conseguenza la scelta del modello di sviluppo implica anche una valutazione dei rischi che le diverse alternative presentano, in ottica di *minimizzazione dei rischi.* Analizziamola in relazione ai metodi di sviluppo del software più classici.

3.9.2.1 Modello a cascata

In questo modello assistiamo alla presenza di un "alto rischio" per i nuovi sistemi con requisiti non stabili, mentre possiamo parlare di "basso rischio" per lo sviluppo di sistemi ben specificati e con tecnologia assestata.

3.9.2.2 Modello a spirale di Boehm



Si tratta di un metodo proposto per supportare l'analisi dei rischi. Si presenta come un modello di tipo ciclico, dove il raggio della spirale rappresenta il costo accumulato durante lo svolgimento del progetto.

Ogni ciclo della spirale rappresenta *una fase:* il più interno può essere lo studio di fattibilità, il successivo la definizione dei requisiti, il successivo ancora la progettazione, etc. Non sono definite fasi a priori. Ogni ciclo della spirale passa attraverso i quadranti del piano che rappresentano i seguenti passi logici:

I: determinazione di obiettivi, alternative e vincoli;

II: valutazione di alternative, identificazione e risoluzione di rischi, ad es. sviluppo di un prototipo per validare i requisiti;

III: sviluppo e verifica del prossimo livello del prodotto, che più essere un modello evolutivo, se i rischi riguardanti l'interfaccia utente sono dominanti; a cascata se è l'integrabilità del sistema il rischio maggiore; trasformazionale, se la sicurezza è più importante;

IV: pianificazione della fase successiva, dove si decide se continuare con un altro ciclo della spirale.

Quanto detto non implica che per un ciclo della spirale si adotti un solo modello di sviluppo. Può descrivere uno sviluppo incrementale, in cui ogni incremento corrisponde a un ciclo di spirale. Può descrivere il modello a

cascata (quadranti I e II corrispondenti alla fase di studio di fattibilità e alla pianificazione del progetto; quadrante III corrisponde al ciclo produttivo). Differisce però dagli altri modelli perché considera esplicitamente il fattore *rischio*.

3.9.3 Il nostro caso di studio

Nel nostro caso di studio le principali fasi del rendering sono composte dalla visualizzazione dell'arcata e dalla misurazione della distanza nella mandibola. La visualizzazione di per sé non rappresenta un rischio di errore, l'errore maggiore potrebbe verificarsi nel caso di errata misurazione. Vi sono diversi algoritmi coinvolti nella elaborazione della misurazione.

3.10 Controllo combinazioni

Per controllo delle combinazioni si intende:

- Uso combinato con altri sw/dispositivi proprietari o no
- Sicurezza e performance

Nel caso di ODOUS non si prevedono combinazioni con altri software, se non il sistema operativo.

In tema di sicurezza sono stati predisposti dei meccanismi di verifica della licenza software, ma non sono previsti meccanismi di protezione dei dati.

4 Bibliografia

[1] Beck K. "*eXtreme Programming Explained*" Addison-Wesley

[2] Beck K. "*Test Driven Development*" Addison-Wesley

[3] Beck K., Beedle M., Bennekum A., Cockburn A., Cunningham W., Fowler M., Grenning J., Highsmith J., Hunt A., Jeffries R., Kern J., Marick B., Martin R., Mellor S., Schwaber K., Sutherland J., Thomas D. "*Manifesto for Agile Software Development*"

[4] Beizer B. "*Design for Testability in Object-Oriented Systems*" COMMUNICATIONS OF THE ACM, Vol. 37, n. 9, Settembre 1990

[5] Boehm B.W. "*Software Engineering Economics*" Prentice Hall

[6] Brooks F.P. "*The Mythical Man-Month: Essays on Software Engineering*" Anniversary Edition, Addison-Wesley

[7] DeMarco T. "*Controlling Software Projects – Management, Measurement & Estimation*" Yourdan Press

[8] Endres A. "*An analysis of errors and their causes in system programs*" ACM SIGPLAN Notices, Vol. 10, n. 6, Giugno 1975

[9] Fowler M., Beck K., Brant J., Opdyke W., Roberts D. "*Refactoring: Improving the Design of Existing Code*" Addison-Wesley

[10] Marco Viceconti, Cinzia Zannoni, Debora Testi, Marco Petrone, Stefano Perticoni, Paolo Quadrani, Fulvia Taddei, Silvano Imboden, Gordon Clapworthy *"The multimod application framework: A rapid application development tool for computer aided medicine"* Da "COMPUTER METHODS AND PROGRAMS IN BIOMEDICINE" 85 (2007) http://www.intl.elsevierhealth.com/journals/cmpb

[11] Gremillion L.L. "*Determinants of Program Repair Maintenance Requirements*" COMMUNICATIONS OF THE ACM, Vol. 27, n. 8, Agosto1984

[12] IEEE: IEEE "*Standard for Glossary of Software Engineering Terminology*" IEEE Std. 828, 1983

[13] Jones C. "*Software assessments, benchmarks, and best practices*" Addison-Wesley

[14] Marchesi M., Succi G. "*Extreme Programmino and Agile Processes in Software Engineering*" Springer

[15] Myers G.J. "*The Art of Software Testing*" John Wiley & Sons

[16] Nielsen J. "*Usability Engineering*" Morgan Kaufmann

[17] Ian Sommerville "*Software Engineering*" Pearson Education, Addison-Wesley

[18] Ghezzi Carlo, Jazayeri Mehdi, Mandrioli Dino "*Ingegneria del software. Fondamenti e principi*", Pearson Education Italia

[19] Stephen Asbury, Peter Ashwell "*Health & Safety, Environment and Quality Audits*" BUTTERWORTH HEINEMANN

[20] "*DIRETTIVA 93/42/CEE DEL CONSIGLIO del 14 giugno 1993 concernente i dispositivi medici*" http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=CONSLEG:1993L0042:20071011:IT:PDF