Provincia di Pavia

Università degli Studi di Pavia

Dipartimento di
Informatica e Sistemistica

DEFINIZIONE DI UN FORMATO DATI STANDARD PER LA TRASMISSIONE DI DATI NELL'AMBITO CALZATURIERO

Bozza 0.7.0

Indice

Introduzione

Scopo del documento

Il presente documento si colloca nell'ambito del progetto "Sostegno all'Innovazione Tecnologica delle PMI in provincia di Pavia: azioni di sistema", che vede la partecipazione di Assomac, dei Dipartimenti di Informatica e Sistemistica e di Meccanica Strutturale dell'Università degli Studi di Pavia, e dell'Agenzia per lo Sviluppo Territoriale, per la definizione di un formato per l'interscambio di dati tra i vari attori del processo progettuale e produttivo calzaturiero.
A seguito di una serie di colloqui con aziende rappresentative del settore sono emersi numerosi problemi, a volte contrastanti, per le quali il formato dati unico vuole proporsi come soluzione.
Questo documento intende essere una proposta concreta di formato dati, e come tale mira a suggerire le caratteristiche essenziali dello standard. In particolare, questo documento copre i trasferimenti dati tra un generico sistema CAD 2D per la progettazione calzaturiera e le macchine da taglio a valle nel processo produttivo.

Sviluppi futuri

Il presente documento rappresenta una prima tappa significativa all'interno di un progetto ampio ed in evoluzione.
Gli standard proposti qui sono da considerarsi stabili nella struttura, almeno per quanto riguarda l'ambito 2D, ma non completi. In particolare, è prevedibile un'evoluzione che porterà all'aggiunta di altre parti dello standard, ora assenti. Si cercherà sempre però di mantenere compatibilità con le versioni mano a mano rilasciate, in modo da non sprecare il lavoro che è stato fatto finora e verrà fatto dai nostri partner aziendali.
In particolare, ecco un elenco delle possibili aggiunte ed evoluzioni che sono prevedibili in questo momento e che verranno realizzate nel prossimo futuro:

- `general.xml` si evolverà in modo da comprendere dei riferimenti ai dati contenuti negli altri file;
- verranno aggiunti altri schemi per formalizzare vincoli di unicità interfile;
- l'attuale proposta per il file `3d.xml` si evolverà in base ad una serie di colloqui che si stanno svolgendo al momento del rilascio del documento per meglio coprire il

trasferimento dati tra ambito di progettazione delle forme e progettazione della calzatura, oltre che l'ambito delle lavorazioni non planari;

- verranno sviluppati degli strumenti software utili per la transizione all'uso del nuovo standard ed il suo consolidamento.

Elementi del processo progettuale e produttivo

Il processo che porta dall'ideazione di un nuovo modello di calzatura alla produzione dell'oggetto finito è costituito da numerose fasi, durante le quali si pone il problema di descrivere i dati in modo che questi siano utilizzabili dagli attori del processo.

La proposta di standard CWA15043, definita nel luglio del 2004 nell'ambito del progetto EFNET 3, ha come obiettivo la definizione di un formato onnicomprensivo per la descrizione dei dati nell'ambito del processo progettuale. Tale proposta tuttavia non ha raggiunto lo status di "standard", e sembra non contare significative implementazioni.

Tra i problemi rimasti aperti sono di particolare rilevanza quelli relativi agli scambi dati geometrici e di processo tra CAD e macchine, e in particolare:

- scambi tra CAD 2D e macchine di taglio;
- scambi tra CAD 3D e macchine per la tornitura delle forme;
- scambi tra CAD 3D e macchine per la preparazione e il montaggio delle varie componenti;
- scambi dati per la definizione delle varianti di materiale e degli ordini di produzione.

Tra questi il più sentito è senz'altro il primo, e per supplire alla mancanza di un formato unico di scambio dati i produttori hanno ripiegato su dialetti del formato DXF, tra loro incompatibili e che comunque soffrono delle limitazioni di un formato non concepito per questi usi.

Il presente documento introduce una proposta di formato dati che:

- rappresenta i dati geometrici 2D senza soffrire delle limitazioni del formato DXF, e in particolare consente descrizioni di tipo vettoriale e semantico laddove appropriate;
- descrive le varianti materiale e gli ordini di produzione;
- descrive le direttive di cucitura delle componenti della scarpa;
- definisce in modo esplicito il significato dei dati, così da non rendere necessaria l'introduzione di dialetti o varianti;
- consente di uniformare la comunicazione tra CAD e macchine di lavorazione e, in misura minore, introduce un formato utile per l'interoperabilità tra diversi CAD;

- rappresenta i dati geometrici 3D in modo da consentirne un facile uso da parte dei CAD e delle macchine.

Questo documento descrive dati che sono indipendenti dalle tecnologie utilizzate e dai dettagli implementativi delle singole macchine o dei singoli CAD. È compito di chi implementa il formato introdurre le componenti software necessarie al fine di consentire al proprio prodotto l'uso di questo formato.

Scopo principale di questo documento è quello di fornire strumenti, non regole. Tuttavia lo standard di comunicazione non sarebbe completo se non contenesse anche le opportune politiche di utilizzo che, se rispettate, garantiscano la compatibilità tra gli attori. Questo aspetto è coperto dall'Appendice B.

Tutte le misure lineari sono espresse in millimetri, con l'eccezione della taglia per la quale l'unità di misura è specificabile. Le misure angolari sono in gradi sessagesimali.

Tutti i numeri rappresentati devono essere nei limiti dello standard IEEE-754 per la rappresentazione dei numeri in virgola mobile a singola precisione. In conformità con quanto previsto dallo standard XML, il separatore decimale è il carattere punto (.) e non è previsto l'uso di un separatore per le migliaia.

Capitolo 1 – Sintesi dei dati e delle informazioni significative

In sede di progettazione della scarpa sono definite tutte le geometrie relative alle varie componenti della calzatura stessa ed alle quali sono associate determinate fasi di lavorazione. Tali geometrie sono ripetute molte volte, in versioni opportunamente scalate in base al numero e alla calzata (lo sviluppo in taglie è un argomento complesso ed è affrontato con algoritmi diversi – e spesso con approcci molto diversi – da ciascun attore).

Per quanto concerne i processi bidimensionali sono state individuate due famiglie di dati, relative rispettivamente alla geometria e al piazzamento automatico.

I dati di geometria comprendono:

- contorni e percorsi interni;
- azioni: ad ogni contorno o percorso è associata un'azione tipica, come ad esempio il taglio, la necessità di segnare con una penna, la necessità di incidere. Va qui notato come durante la fase di descrizione della geometria non si possano né si debbano fare ipotesi sulla metodologia con la quale verrà attuata l'azione specificata, in quanto dipendente fortemente dalla tecnologia adottata;
- testi, sia da stampare sul pezzo sia da accompagnare ad esso (ad esempio, può essere stampato su carta da una stampante integrata nella macchina);
- tacche di riferimento: queste tacche, di varia foggia, sono poste lungo il perimetro di un pezzo e servono come riferimento per l'allineamento dei pezzi stessi;
- tacche simboliche: sono tacche a cui si dà il significato semantico di segnalare il numero del pezzo, cioè la taglia della scarpa a cui appartiene quella geometria;
- destro/sinistro: informazioni che specificano se le varie geometrie si riferiscono alla scarpa destra o sinistra (l'altra è di norma ottenuta per ribaltamento, ma è possibile definire paia di scarpe non speculari).

In alcuni casi può essere opportuno fornire indicazioni relative alle politiche da seguire per tener conto dell'ingombro utensile.

I dati relativi al piazzamento automatico sono necessari solo per quelle macchine da taglio che posizionano automaticamente i pezzi sul materiale da lavorare (dopo averne acquisite le caratteristiche attraverso una scansione) e che quindi non necessitano dell'ausilio di un operatore. Tali dati comprendono:

- direzionalità del pezzo: è un vettore direzione che indica come va posizionato il pezzo rispetto alla direzione di riferimento della pelle (ad esempio: rispetto alla direzione testa-coda). Va inoltre specificata una tolleranza su tale direzionalità;
- aree/contorni di qualità: alcune zone del pezzo possono tollerare una qualità meccanica e/o estetica inferiore, mentre altre necessitano della migliore qualità disponibile. La pelle è per sua natura irregolare (sono presenti graffi, tagli e difetti di diverso tipo e gravità) ed il piazzamento automatico deve tenerne conto. Questo dato potrebbe comunque essere di una qualche utilità anche per l'utente che si occupa del piazzamento manuale.

Ricordiamo inoltre che spesso le informazioni relative al piazzamento automatico non sono di competenza del progettista, che si occupa di disegnare lo stile della scarpa ma che non decide il materiale, anche se è in fase di progettazione che si stabilisce quali componenti della scarpa abbiamo minore o maggiore rilevanza (visibilità) nel prodotto finito e quindi necessitino di essere tagliati su porzioni di pelle di adeguata qualità.

Relativamente ai dati 3D le informazioni significative sono relative a:

- geometria della forma, separata nelle 3 superfici canoniche (top, upper e bottom);
- individuazione delle aree di lavorazione;
- elementi principali della forma di soletta e tacco.

Per quanto concerne le relazioni, i dati che occorre trasferire comprendono:

- assemblaggi: quali pezzi vanno cuciti assieme e come;
- materiali: quale specifico materiale va usato per ogni pezzo. A livello di progettazione si sa che un pezzo sarà, ad esempio, in pelle. A livello di produzione si specifica il tipo particolare di pelle;
- ordini di produzione: quanti pezzi bisogna tagliare (ovvero quante scarpe produrre).

Capitolo 2 – Strutturazione generale del file di scambio dati

Gli scambi dati che il formato in oggetto si propone di agevolare coprono informazioni di diversa natura e, in diversi casi, introdotti da soggetti diversi. Per conciliare i problemi di strutturazione e separazione dei dati con la richiesta di avere tutte le informazioni in un singolo file introduciamo il concetto di archivio. Con questo termine indichiamo un singolo documento che internamente è suddiviso in file e cartelle, e che è ragionevolmente implementabile utilizzando il formato di archiviazione ZIP. L'estensione prevista da assegnare all'archivio è "`.jane`", in quanto si vuole evitare che l'archivio venga decompresso e modificato per errore da chi lo maneggia (il cambiamento di estensione solitamente inibisce dall'apertura accidentale attraverso programmi di archiviazione). I file presenti all'interno dell'archivio saranno di tipo XML 1.0.
La struttura dei file all'interno dell'archivio è la seguente:

```
/ (root)
|
+- META-INF (cartella)
|   |
|   +- manifest.xml
|
+- EXTRA (cartella)
|
+- general.xml
|
+- 2d.xml
|
+- assembly.xml
|
+- 3d.xml
|
+- materials.xml
|
+- production.xml
```

Segue una breve descrizione del contenuto di ciascun file:

- `META-INF`

  cartella il cui scopo è quello di contenere i file che trasportano informazioni generali relative al tipo di file;
- `EXTRA`

  cartella contenente eventuali file supplementari (fogli di stile, immagini, ...) necessari ai documenti da stampare in fase di produzione;
- `manifest.xml`

  come è d'uso in documenti strutturati come archivio, il file manifest fornisce

informazioni circa il tipo di file (ad esempio, il numero di versione dello standard utilizzato);

- `general.xml`
  questo file conterrà informazioni di carattere generale in merito al progetto in questione (tra cui, ad esempio, il nome del modello base e della variante);
- `2d.xml`
  in questo file sono contenute tutte le informazioni relative alle geometrie bidimensionali delle scarpe, affiancate alla distinta base neutra (numero di istanze per ciascun componente della scarpa e classe di materiale);
- `assembly.xml`
  qui vengono memorizzate le informazioni relative alle cuciture e, più in generale, all'assemblaggio planare della scarpa;
- `3d.xml`
  qui vengono memorizzate le geometrie della forma e vengono inoltre individuati un sistema di riferimento e le aree di lavorazione;
- `materials.xml`
  in questo documento sono descritte tutte le combinazioni di materiale usate per il modello di scarpa;
- `production.xml`
  sono qui annotate le informazioni circa il numero di scarpe da produrre per ciascuna variante.

Solo i file `manifest.xml` e `general.xml` sono obbligatori: dalla combinazione di alcuni o tutti gli altri file cambia il significato e l'uso che si può fare dell'intero archivio. A titolo esemplificativo:

- tutti i file: questo è un progetto completo, che raccoglie tutte le informazioni che toccano i vari aspetti della produzione;
- presenti `2d.xml`, `assembly.xml`, `materials.xml`, `production.xml`: l'archivio può essere inviato dal settore progettazione al settore produzione, in quanto contiene tutte le informazioni necessarie (riguardo alla produzione bidimensionale, quindi principalmente legata al taglio);
- solo `production.xml`: l'archivio è simile ad una bolla di produzione, e viene utilizzato nel caso in cui il produttore debba eseguire un ordine appartenente ad un

progetto già noto, perché comunicato precedentemente (ad esempio con un archivio come quello descritto al secondo punto).

È prevista l'introduzione di un meccanismo che consenta di associare automaticamente due o più file `.jane` relativi al medesimo progetto.

Capitolo 3 – Introduzione al formato

Quanto segue descrive il contenuto dei file interni all'archivio. La descrizione è informale, ma rappresenta quella che dovrebbe essere la struttura dei file XML previsti. Una descrizione formale è fornita nel Capitolo 4.

In numerosi punti dei file sono indicati tag `<description field="..." value="..."/>`, il cui scopo è quello di rappresentare commenti che, in generale, non devono essere interpretati per il completamento delle fasi di lavorazione. Alcuni campi (come "`author`") sono standardizzati per consentirne ad esempio la visualizzazione automatica (si veda in proposito l'Appendice A), ma è prevista la possibilità di aggiungere descrizioni in campi con nomi arbitrari. Si è preferito utilizzare un solo tipo di tag ("`description`") con un campo che ne specifica il tipo piuttosto che numerosi tag specializzati per semplificare l'interpretazione – ciò consente infatti ad un programma di visualizzare tutte le informazioni contenute senza supportare lo specifico campo e distinguendole da dati di natura diversa.

Manifesto (`manifest.xml`)

Scopo primario di questo file è riportare informazioni sul tipo di documento e sulla versione di standard alla quale questo fa riferimento.

Il formato del file è il seguente:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<manifest>
  <fileEntry mediaType="assomac/jane" versionString="0.7.0"/>
</manifest>
```

dove la stringa "`0.7.0`" identifica la versione dello standard implementata.

Informazioni generali (`general.xml`)

Il formato del file è il seguente:

```
<rootGeneral>
  <description field="author" value="Nome dell'autore"/>
  <description field="baseModel" value="Nome modello base"/>
  <description field="variant" value="Nome variante"/>
  {... altre descrizioni ...}
  <info id="Informazioni aggiuntive">
    <![CDATA[
      <?xml version="1.0" encoding="ISO-8859-1"?>
      <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
                            "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
      <html>
        <head>
          <title>Informazioni extra sul progetto</title>
        </head>
        <body>
          {... testo contenente le informazioni, formattato XHTML 1.0 Strict ...}
        </body>
      </html>
    ]]>
  </info>
</rootGeneral>
```

Il file di generalità contiene informazioni di valore complessivo per il progetto. Questo attualmente include solo descrizioni circa il modello di scarpa rappresentato, l'autore e in generale informazioni testuali che non vengono elaborate.

Possono essere inoltre specificate informazioni aggiuntive utilizzando il tag `info` (opzionale), che consente di includere all'interno del file XML informazioni aggiuntive ed anche formattazione grafica: è infatti possibile includere qui un documento XHTML 1.0 Strict[1] (che non deve contenere alcun tipo di script). Eventuali file accessori (immagini, ...) possono essere memorizzati nella cartella `EXTRA` dell'archivio. Come si vedrà nel seguito, questo tag è utilizzabile anche in altri file, mantenendo comunque la struttura qui descritta.

1 Il documento complessivo deve superare il test formale fornito dal W3C, http://validator.w3.org/

Geometrie bidimensionali (`2d.xml`)

Il formato del file è il seguente:

```
<root2d>
  <componentDependent id="1" materialClassId="23" rightCount="1" leftCount="1"
                      side="left/right" direction="45" directionTolerance="10">
    <description field="name" value="Tomaia I"/>
    {... altre descrizioni ...}

    <slicing depth="10" lift="0.3" angle="20" visibleSide="yes/no"
                                              folding="none/simple/reinforced">
        <slicedTract id="0"/>
        {... altri tratti di perimetro da scarnire in questo modo ...}
    </slicing>
    {... altre scarniture di tipo diverso ...}
  </componentDependent>
  {... altre intestazioni di componente ...}

  <geometryDependent id="3142" sizeUnit="eu" sizeValue="37" footLength="370"
                                       toeWidth="10" foreFit="22" heelFit="20"/>

    <component id="1">
      <perimeter/closedPath/openPath/area id="1"
             action="cut/etch/mark/show/quality" toolSide="0"
                      mechanicalQuality="high/low" aestheticQuality="high/low">

        <point x="0" y="10"/>
        <tract id="0" smoothness="high/low" fidelity="high/low" maxToolSize="2">
          <sampled/vector/...>
            {... direttive dipendenti dal tipo di descrizione ...}
          </sampled/vector/...>
          {... altre descrizioni, di altro tipo ...}
        </tract>
        <point x="10" y="10" flags="smooth/sharp"/>
        <tract id="1" smoothness="high/low" fidelity="high/low">
          <sampled/vector/...>
            {... direttive dipendenti dal tipo di descrizione ...}
          </sampled/vector/...>
          {... altre descrizioni, di altro tipo ...}
        </tract>
        {... coppie punto-tratto completanti il percorso ...}
      </perimeter/closedPath/openPath/area>
      {... altri percorsi per questo componente ...}
    </component>
    <component id="2">
      <sameAsGeometry id="1"/>
    </component>
    {... altri componenti per questa taglia ...}
  </geometryDependent>
  {... geometrie per taglie diverse ...}
  <info id="Informazioni aggiuntive">
    {... informazioni aggiuntive ...}
  </info>
</root2d>
```

La definizione delle geometrie 2D inizia con una sequenza di tag `componentDependent`, nei quali vengono descritte le informazioni (indipendenti dalla taglia) relative ad uno specifico pezzo della scarpa. L'identificatore (`id`) del componente è una stringa che consente un

riferimento univoco, usato nel seguito quando verranno definite le geometrie (non sono quindi ammissibili due `componentDependent` che condividano lo stesso `id`).

Come attributi del tag `componentDependent` sono inoltre specificati i seguenti dati:

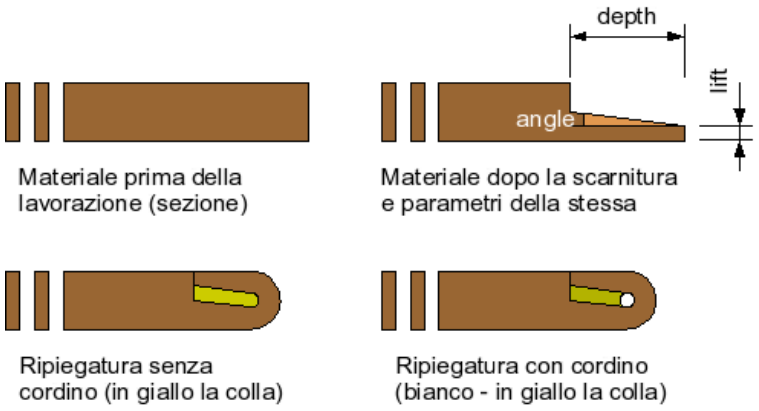
- `materialClassId`: è l'identificativo (`id`) della classe del materiale da utilizzare per realizzare quello specifico componente. Tale valore verrà accoppiato ad uno specifico materiale reale all'interno del file `materials.xml`
- `rightCount`/`leftCount`: quante istanze di quel componente occorrano per la fabbricazione di un singolo pezzo della scarpa, sia per il lato descritto (considerando che di norma si descrive solo la scarpa destra o solo la sinistra), sia per quello speculare. In questo modo è possibile anche la descrizione di paia di scarpe non speculari.
- un lato (`side`), che consente di capire se la descrizione geometrica fornita è relativa alla scarpa destra o a quella sinistra;
- una direzione (`direction`), che identifica un versore direzione associato al pezzo. La direzione di riferimento delle pelli è determinata a priori, ad esempio si può usare quella testa-coda;
- una tolleranza sulla direzione (`directionTolerance`), che segnala di quanto è lecito scostarsi, in fase di piazzamento, dalla direzione imposta (il pezzo può essere posizionato con uno scostamento di ±`directionTolerance` sulla direzione ottimale).

Nello schema è presente un tag `description` per specificare il nome del pezzo: sebbene sia opzionale, è particolarmente utile per orientarsi all'interno di un progetto di calzatura complesso.

Il tag `componentDependent` si chiude con un elenco di lavorazioni che sono dipendenti dal componente ma non dalla taglia. In particolare attualmente l'unica lavorazione di questo tipo individuata è la scarnitura/ripiegatura.

La scarnitura consiste nell'assottigliamento del materiale sul perimetro, caratterizzato dai seguenti parametri (si faccia riferimento alla figura):

- profondità (`depth`) della lavorazione;
- altezza (`lift`) dell'utensile e sua inclinazione (`angle`);
- la necessità di scarnire dal lato visibile (`visibleSide="yes"`) oppure (come più frequente) da quello non visibile (`visibleSide="no"`);



Scarnitura e ripiegatura

mentre la ripiegatura, la cui entità è determinata dalla profondità della scarnitura, è caratterizzata dalla presenza o meno del cordino di rinforzo (l'attributo `folding` vale `"none"` se il tratto non è da ripiegare, `"simple"` se è da ripiegare senza cordino, `"reinforced"` se invece occorre il cordino di rinforzo).

Considerato che molti (se non tutti) i tratti da scarnire richiederanno la medesima lavorazione, le caratteristiche di questa sono raggruppati nel tag `slicing`, il cui contenuto diventa solo l'elencazione di uno o più tratti di perimetro (tag `perimeter`, descritto più avanti) che occorre scarnire in tale maniera specificati con l'attributo `id` del tag `slicedTract`. È possibile specificare diversi parametri di scarnitura semplicemente usando più tag `slicing` in successione.

L'elenco delle proprietà delle componenti è seguita dalla descrizione delle geometrie (tag `geometryDependent`). L'identificatore (`id`) delle geometrie è una stringa univoca che la identifica all'interno del file, e all'interno di ciascuna geometria è completamente descritta l'intera scarpa per una specifica taglia. Come identificatore è pertanto possibile utilizzare un'espressione derivante più o meno direttamente dalla taglia della scarpa, ma non si tratta di un obbligo.

L'associazione della geometria a una taglia specifica è fatta con i successivi attributi di `geometryDependent` (le "proprietà dimensionali"), che rappresentano le seguenti misure:

- `sizeValue` e `sizeUnit` rappresentano rispettivamente la taglia della calzatura e l'unità di misura in cui è espressa, da scegliere tra quelle elencate nella tabella "Unità di misura della taglia".

Codice	*Descrizione*
au-m / au-w	Australia – scale uomo / donna
en	EN 13402
eu	Europa
jp-m / jp-w	Giappone – scale uomo / donna
mp	Mondopoint / Korea / ISO 9407
mx	Messico
us-m / us-w / us-b / us-g	USA & Canada – scale uomo / donna / bambino / bambina
uk-m / uk-w / uk-b / uk-g	Regno Unito – scale uomo / donna / bambino / bambina

Unità di misura della taglia

Si tenga presente che l'indicazione della taglia non serve ad esprimere se la scarpa è, ad esempio, da uomo o da donna, anche perché potrebbero sorgere problemi nel caso di modelli unisex. Si tratta semplicemente di un'unità di misura.

L'utilizzo di questi due campi è obbligatorio, e possono essere usati per fornire ad un operatore una misura "leggibile" della dimensione della scarpa;

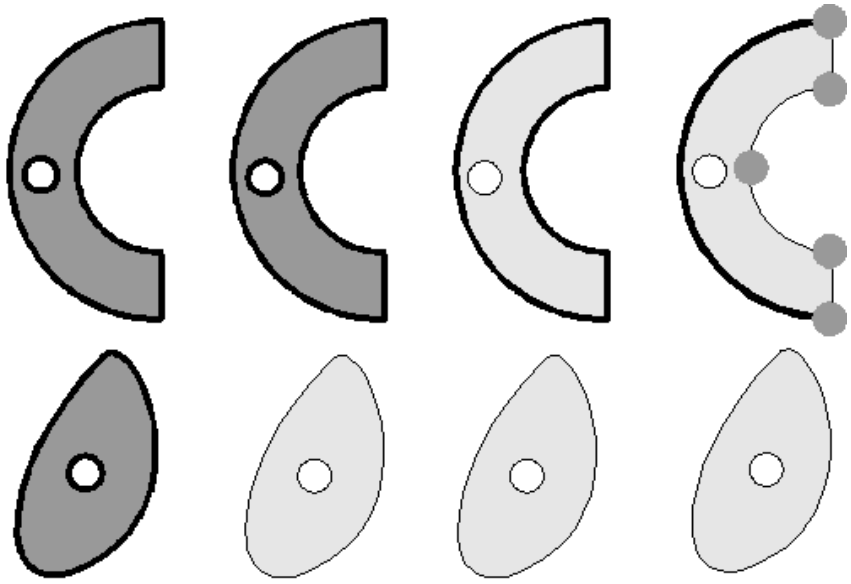
- `footLength` esprime la misura della lunghezza del piede calzato dalla scarpa (obbligatorio, in millimetri);
- `toeWidth` è la misura della larghezza del piede all'altezza delle dita (opzionale, in millimetri);
- `foreFit` è la calzata della punta (opzionale, in millimetri);
- `heelFit` è la calzata del tallone (opzionale, in millimetri).

La descrizione geometrica di ciascuna componente necessaria per la taglia in questione è data nel tag `component`, ed è caratterizzata da un identificatore (`id`), che richiama l'identificatore del tag `componentDependent` corrispondente.

Il contenuto del tag `component` può essere di due tipi. Se il componente che si vuole rappresentare è identico all'omologo usato per un'altra taglia, nella quale questo è stato definito, allora è possibile riciclare quanto già descritto usando il tag `sameAsGeometry`, il cui attributo `id` fa riferimento all'identificatore della sezione `geometryDependent` che contiene la definizione del componente. In tal caso il tag `component` si chiude subito dopo `sameAsGeometry`.

In alternativa occorre procedere alla descrizione completa del componente.

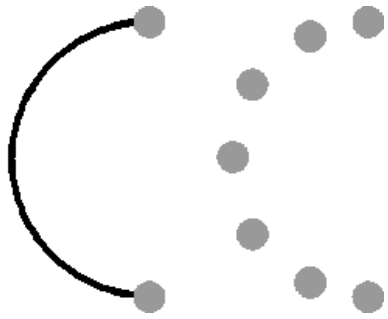
Le componenti sono suddivise in percorsi, che sono l'unità più generale per il disegno e rappresentano linee aperte, linee chiuse oppure aree. I percorsi sono a loro volta suddivisi in tratti (`tract`), che curano la descrizione di specifici sottoinsiemi del percorso. Per meglio chiarire il concetto si faccia riferimento alla figura sotto, in cui la parte *a)* rappresenta una geometria completa (`geometryDependent`), *b)* un componente (`component`) della geometria, *c)* un percorso (in questo caso un `perimeter`) e *d)* un tratto (`tract`) con evidenziati gli estremi di tutti i tratti di quel percorso.



a) *b)* *c)* *d)*

Scomposizione delle geometrie

Per ottenere flessibilità e estensibilità del formato, oltre che per ammettere l'implementazione di sottoinsiemi del formato, ciascun tratto può essere descritto in più modi. Per fissare le idee si osservi la figura seguente, nella quale il medesimo tratto è rappresentato con un arco di cerchio (*a)*) e con una campionatura (*b)*).



a) *b)*

Diversi modi di rappresentazione di un tratto

Sono definiti 3 tipi di percorso:

- percorsi chiusi che corrispondono a una lavorazione (`perimeter`, che corrisponde al perimetro del componente e come tale deve comparire una e una sola volta, e `closedPath`);
- percorsi aperti che corrispondono a una lavorazione (`openPath`);
- percorsi chiusi che delimitano il bordo esterno di un'area (`area`).

Un percorso è caratterizzato da un identificativo (`id`), univoco all'interno del `component`, che consente di far riferimento al percorso da altre parti del documento, da un'azione (`action`), che riporta un'informazione semantica relativa al percorso (la sua natura), e da eventuali altre proprietà secondo quanto descritto nel seguito.

Le azioni previste dallo standard sono le seguenti:

- "`cut`" (taglio)
- "`etch`" (incisione)
- "`mark`" (segnato)
- "`quality`" (area di qualità)
- "`show`" (visualizzazione delle linee a uso dell'operatore, alla quale non corrisponde alcuna lavorazione fisica).

L'associazione delle azioni ai percorsi non è completamente libera. La tabella seguente mostra quali azioni sono associabili a quali percorsi.

		Azione				
		cut	**etch**	**mark**	**quality**	**show**
Percorso	**perimeter**	✓				
	closedPath	✓	✓	✓		✓
	openPath	✓	✓	✓		✓
	area				✓	✓

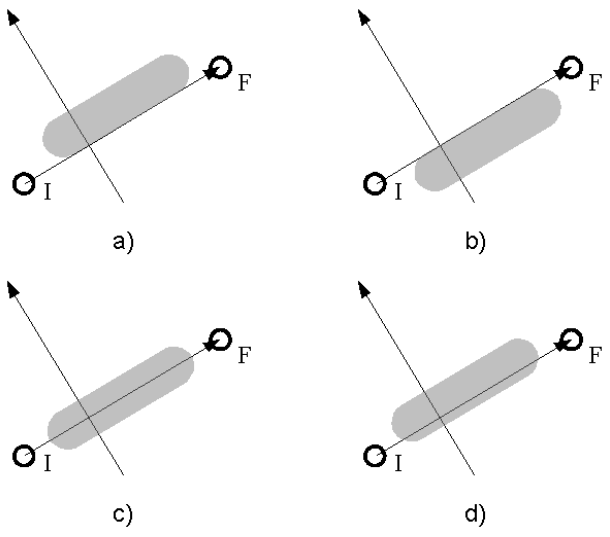
Coppie percorso–azione lecite

Ci sono ancora due attributi da descrivere:

- `mechanicalQuality` e `aestheticQuality`, necessari (e leciti) solo quando l'azione è `quality`, che specificano i livelli di qualità meccanica (resistenza) ed estetica associati al percorso. Visto che solo i percorsi di tipo `area` possono avere l'azione `quality`, com'è possibile intuire questo è il modo di esprimere la qualità dell'area racchiusa dal percorso. I livelli di qualità sono stringhe che possono valere `high` (alta qualità) oppure `low` (bassa qualità).

  Nel caso di più aree di qualità sovrapposte, quella definita successivamente ha priorità sulle precedenti;
- `toolSide`, valido e necessario solo con azione `cut`, `mark` o `etch`, che fornisce un'indicazione su come gestire lo spessore dell'utensile. Con riferimento alla figura seguente, si pensi al caso di un taglio interno con un'utensile di spessore non trascurabile. Con `toolSide="0"` l'utensile si posizionerà sopra la linea e asporterà

Legenda:
- ✗ I: punto di inizio
- ✗ F: punto di fine
- ✗ caso a): `toolSide` = 1
- ✗ caso b): `toolSide` = -1
- ✗ caso c): `toolSide` = 0
- ✗ caso d): `toolSide` = 0.5



Utilizzo del parametro "`toolSide`"

un'area eccedente il profilo indicato simmetricamente da entrambi i lati. Utilizzando `toolSide="1"` invece l'intera area asportata giacerà alla sinistra della linea indicata, così come con `toolSide="-1"` questa giacerà completamente alla sua destra. È possibile utilizzare qualsiasi valore tra -1 e 1 compresi.

Il tag `perimeter` deve comparire una ed una sola volta per ogni componente.

Il contenuto dei tag `perimeter`, `openPath`, `closedPath` e `area` consiste nella definizione geometrica del percorso.

I percorsi sono in generale suddivisi in N tratti differenti, i cui estremi sono ottenuti specificando le coordinate di N punti (se si tratta di un percorso chiuso) ovvero quelle di N+1 punti (se il percorso è aperto). I punti e i tratti si alternano nella definizione, così che un tratto sia sempre preceduto e seguito dai punti che identificano il suo inizio e la sua fine, con l'eccezione dell'ultimo tratto di un percorso chiuso, che non è seguito da un punto perché questo deve coincidere con quello, già specificato, di inizio del primo tratto.

Un punto è caratterizzato da:

- i campi `x` e `y`, che contengono le coordinate *x* e *y* del punto;
- un campo `flags`, opzionale, che contiene ulteriori specifiche del punto. Attualmente sono previsti:
  - `smooth / sharp`, che specificano che le entità geometriche che fanno capo al punto si uniscono senza formare / formando un punto angoloso.

Un tratto è caratterizzato da:

- un identificativo (`id`), che deve essere univoco all'interno del percorso (`openPath`, `closedPath`, `area`, `perimeter`) e il medesimo per i tratti omologhi in geometrie diverse;
- un indicatore (`smoothness`) che determina la cura con la quale deve essere eseguita l'operazione associata (`smoothness="high"` è sempre la massima qualità possibile, mentre `"low"` indica che la qualità può essere abbassata). Ad esempio, nel caso del taglio potrebbe essere ammissibile che i lembi siano "ruvidi";
- un indicatore (`fidelity`), che indica se la geometria del tratto debba essere rispettata fedelmente oppure no (`fidelity="high"` indica che il percorso deve essere seguito fedelmente, mentre il valore `"low"` autorizza il sistema a "spostare" leggermente la curva qualora questo risultasse vantaggioso);

- una massima dimensione dell'utensile (`maxToolSize`), che rappresenta la massima larghezza complessiva accettabile per un utensile che deve fare quella lavorazione (un esempio in cui questa indicazione può essere utile è quello del taglio di un profilo usando fustelle circolari o frese). Se non specificato, si assume che non esista limite superiore significativo alla dimensione dell'utensile;
- una o più definizioni geometriche che consentano di coprire la distanza che separa gli estremi del tratto.

Sono attualmente previste le seguenti rappresentazioni:

Descrizioni campionate

Le descrizioni campionate constano di un elenco ordinato di punti, che l'utilizzatore è libero di interpolare nella maniera ritenuta più opportuna. Un tag `<break/>` segnala che tra il punto precedente e quello successivo la curva è interrotta (essenzialmente occorre "saltare" dal punto precedente al successivo).

Descrittore:

```
<sampled>
  <point x="3" y="2" flags="smooth"/>
  {... ...}
  <break/>
  {... ...}
</sampled>
```

Descrizioni vettoriali

Una descrizione vettoriale, racchiusa all'interno del tag `<vector>`, consta di un'alternanza di punti e curve, che inizia e termina sempre con una curva. Il punto di inizio delle curve coincide sempre con il punto che ne precede la definizione, mentre il punto di fine coincide col punto che la segue (la prima curva inizia dal primo estremo del tratto, mentre l'ultima termina nel secondo estremo dello stesso); per questo nelle definizioni che seguono i punti di inizio e fine sono sempre omessi.

Le curve previste sono le seguenti:

Cerchio

La direttiva geometrica cerchio permette di tracciare una circonferenza. I due punti di inizio e fine rappresentano gli estremi di un diametro e quindi individuano anche centro (punto medio) e raggio (metà della distanza).

Si assume, ai fini dell'interpretazione del parametro `toolSide`, che il cerchio venga percorso in senso antiorario (quindi impostando `toolSide="1"` impone che l'utensile stia completamente al suo interno).

Descrittore:

```
<circle/>
```

Arco

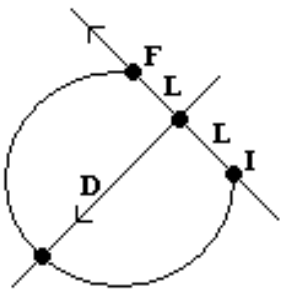
Un arco di cerchio è una direttiva geometrica che permette di congiungere i due punti con una parte di circonferenza, che viene identificata univocamente usando un punto supplementare, identificato da un indice di rigonfiamento ("bulge") dell'arco.

Si assuma che l'arco da rappresentare sia quello disegnato in figura. L'asse del segmento che unisce i punti di inizio e di fine interseca l'arco in un punto. Il bulge è definito come il rapporto D/L.

Utilizzare archi a bulge nullo è un modo per realizzare segmenti dritti.

Descrittore:

```
<arc bulge="1"/>
```



Legenda:

- I: punto di inizio
- F: punto di fine
- L: semilunghezza del segmento I-F
- D: spiazzamento del terzo punto dal segmento I-F

Esempio di arco

Curva di Bézier

Questa direttiva permette di definire curve attraverso un calcolo polinomiale. È possibile specificare i punti di controllo, tenendo presente che il primo e l'ultimo (estremi della curva) sono già implicitamente specificati.

L'utilizzo di una curva di Bézier senza punti di controllo è un modo lecito per generare un segmento di retta.

Descrittore:

```
<bezier>
  <bezierControlPoint x="3" y="2"/>
  {... ...}
```

```
</bezier>
```

Salto

Un salto è una direttiva geometrica che segnala la necessità di raggiungere il punto successivo senza compiere alcuna operazione. Tale direttiva è necessaria per definire geometrie discontinue, come il testo (quando occorra descriverlo in modo vettoriale, ad esempio perché occorre che sia realizzato con uno stile particolare), o per realizzare trame come, ad esempio, linee di fori.

Non sono specificati altri parametri.

Descrittore:

```
<break/>
```

Descrizioni semantiche

Le descrizioni semantiche consentono di specificare il significato di alcuni tipi di percorso, ad uso sia della macchina sia di un eventuale CAD, nell'ottica di ottenere interoperabilità orizzontale. Sono attualmente previsti i seguenti modi semantici:

Testo

Questo modo permette di esprimere un testo come successione di caratteri. Scopo di questo modo è comunicare semplicemente il testo (`text`) che deve essere scritto o stampato ed un'area rettangolare entro la quale posizionarlo. Sarà cura di chi esegue la scrittura scegliere la dimensione dei caratteri, la spaziatura ed il posizionamento all'interno dell'area.

Quello del testo semantico, come tutti gli altri modi, è associato ad un tratto, e quindi sono definiti un punto di inizio ed un punto di fine: tali punti si immaginano congiunti da un segmento e definiscono la base del rettangolo che delimita l'area del testo. Nella definizione del modo viene anche specificato un parametro di altezza (`height`), che porta quindi alla completa definizione del rettangolo (il testo giace alla sinistra del vettore identificato dai punti di inizio e fine).

Esempio:

```
<semanticText height="10" text="Queste parole vanno stampate"/>
```

Tacca simbolica

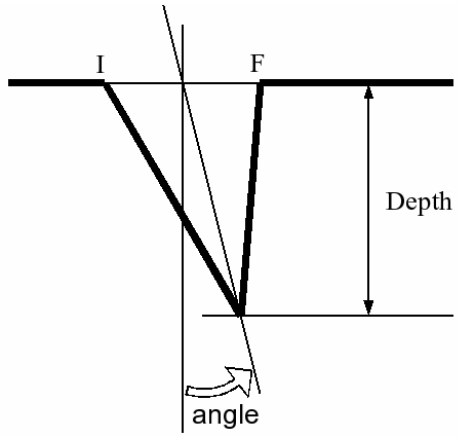
Questo modo specifica che il tratto va colmato con una tacca simbolica che specifica la taglia a cui appartiene il percorso.

L'effettiva forma delle tacche è lasciata al responsabile del macchinario, che può decidere ad esempio se eseguire tacche quadrate o arrotondate, più o meno ampie, ecc. L'unico vincolo è che la tacca simbolica rimanga all'interno dei limiti del tratto.
È presente un parametro per specificare se la tacca debba giacere alla destra o alla sinistra del tratto. Dato che le descrizioni semantiche lasciano un certo gradi di libertà nella realizzazione dell'elemento, e considerato che alcune tacche possono essere bilaterali (es. a volte la tacca del mezzo numero è fatta dal lato opposto alle altre) è presente un parametro ulteriore che autorizza o vieta l'occupazione del lato opposto a quello indicato.
Esempio:

```
<symbolicNotch side="right" allowOpposite="no"/>
```

Tacca di riferimento

Questo modo specifica che il tratto va colmato con una tacca di riferimento che può essere utile, ad esempio, come punto di riferimento durante la fase di montaggio.
L'effettiva forma delle tacche è lasciata al responsabile del macchinario, che però deve rispettare la lunghezza della tacca (coincidente con quella del tratto), la profondità e l'angolo di inclinazione misurato in senso antiorario, specificato secondo quanto riportato in figura.
È presente un parametro per specificare se la tacca debba giacere alla destra o alla sinistra del tratto.



Esempio di tacca di riferimento

Esempio:

```
<referenceNotch side="right" depth="2" angle="20"/>
```

Informazioni aggiuntive

È possibile specificare informazioni aggiuntive, già formattate graficamente per essere stampate o visualizzate, utilizzando il tag opzionale `info`. Per le regole sull'uso di tale tag si vedano i dettagli relativi a `general.xml`

Assemblaggi (`assembly.xml`)

Il formato del file è il seguente:

```
<rootAssembly>
  <geometry id="3">
    <stitching id="1">
      <components>
        <component id="0" flipX="yes/no" rotation="20" offsetX="10" offsetY="-30"/>
        {... altre componenti da cucire assieme ...}
      </components>
      <path id="1">
        <strictSample>
          <point x="2" y="3"/>
          {... altri punti di cucitura ...}
        </strictSample>
        <openStitchPath/closedStitchPath width="0" length="3" swingPoints="0"
                                beginBackTackingCount="2" beginBackTackingLength="3"
                                   endBackTackingCount="2" endBackTackingLength="3">
          <point x="4" y="5" flags="smooth"/>
          <tract id="2">
            <sampled>
              <point x="5" y="8" flags="smooth"/>
              {... altri campioni della linea di base ...}
            </sampled>
            <vector>
              <arc bulge="0"/>
              <point x="5" y="0" flags="sharp"/>
              <bezier></bezier>
              {... altri curve analitiche ...}
            </vector>
          </tract>
          <point x="4" y="5" flags="smooth"/>
        </openStitchPath>
      </path>
      {... altre cuciture per queste componenti ...}
    </stitching>
    {... altre cuciture per questa taglia ...}
  </geometry>
  {... cuciture per altre taglie ...}
</rootAssembly>
```

Il file descrive le cuciture da fare sulle componenti della scarpa, raggruppate per taglia (geometria). L'identificativo (`id`) del tag `geometry` fa riferimento all'identificativo usato nel tag `geometryDependent` del file `2d.xml`, e identifica così univocamente le componenti cui fa riferimento.

Per ciascuna geometria possono essere specificate più cuciture (`stitching`), ciascuna identificata da un identificatore univoco all'interno della geometria.

Le cuciture sono specificate in 2 passi:

- prima vengono specificate le componenti coinvolte assieme ad una trasformazione geometrica che consente di specificarne la posizione relativa;
- poi viene specificato il percorso di cucitura.

Il primo passo può essere immaginato come la descrizione dell'operazione di spostamento dei pezzi dalla posizione nella quale sono stati descritti ad una posizione che consente di cucirli assieme. In altri termini, dopo la trasformazione le componenti si trovano posizionate come risulterebbero su un pallet di cucitura (per quanto le coordinate e l'orientamento globali restino arbitrari).

Le trasformazioni lecite per questa lavorazione sono un sottoinsieme delle trasformazioni affini. In particolare sono consentite:

- traslazione;
- rotazione;
- capovolgimento.

Per garantire il rispetto di questo vincolo la trasformazione è descritta da 4 parametri, che sono:

- capovolgimento, descritto analiticamente da un cambio di segno della coordinata x;
- rotazione di un angolo θ attorno all'origine degli assi in direzione antioraria;
- traslazione di un vettore (δ_x, δ_y).

Le operazioni vengono eseguite nell'ordine indicato nell'elenco, e danno luogo alla trasformazione affine:

$$T = \begin{pmatrix} f \cdot \cos\theta & -\sin\theta & \delta_x \\ f \cdot \sin\theta & \cos\theta & \delta_y \\ 0 & 0 & 1 \end{pmatrix}$$

dove f vale +1 se il pezzo non deve essere capovolto oppure -1 se il pezzo deve essere capovolto. La trasformazione T trasforma le coordinate di un generico punto del componente nello spazio in cui è descritto (x, y) nelle coordinate "di arrivo" $(\hat{x}, \hat{y})$ secondo l'identità

$$\begin{pmatrix} \hat{x} \\ \hat{y} \\ 1 \end{pmatrix} = T \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

La specificazione delle componenti da cucire assieme consta di un elenco ordinato di tag `component` racchiusi in un tag `components`. Per ciascuna componente sono specificate l'identificativo (`id`) del componente nella corrispondente definizione data nel file `2d.xml` e i parametri di posizionamento con la seguente corrispondenza attributo-parametro:

- f vale +1 se e solo se `flipX="no"`;
- θ è specificato dall'attributo `rotation`;
- δ_x corrisponde all'attributo `offsetX`;
- δ_y corrisponde all'attributo `offsetY`.

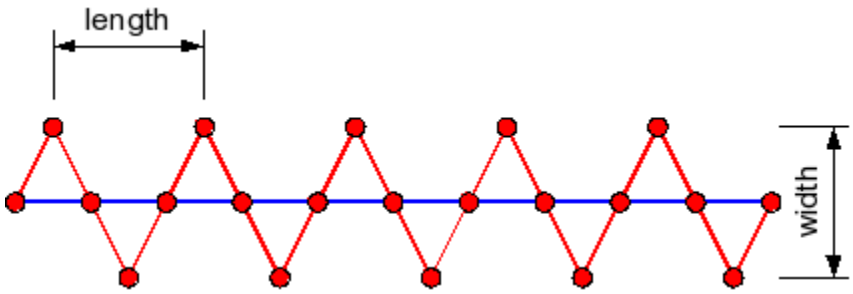
Qualora le componenti, dopo il posizionamento, risultassero sovrapposte (anche solo parzialmente), si assume che queste siano "impilate" dal basso verso l'alto, con il lato a vista rivolto verso l'alto (se `flipX="no"`, verso il basso altrimenti) in modo che il componente specificato prima sia sotto quello specificato dopo.

Dopo aver specificato quali componenti cucire e quale sia la loro posizione relativa vengono specificati uno o più percorsi di cucitura, ciascuno racchiuso in un tag `path`. Esistono più modi per specificare il medesimo percorso, divisi in 2 classi:

- la classe `strictSample` consente di specificare dove devono essere apposti i singoli punti di cucitura (questo può essere utile ad esempio qualora si desiderasse realizzare dei fregi col filo);
- la classe vettoriale, composta dai tag alternativi `openStitchPath` (per le cuciture "aperte") e `closedStitchPath` (per le cuciture "chiuse", cioè cuciture che chiudono un anello), consente di descrivere la cucitura in modo parametrico, lasciando alla macchina l'onere di decidere la posizione dei singoli punti in modo conforme.

Nel primo caso, l'elenco ordinato dei punti in cui far scendere l'ago è fornito elencandone le coordinate all'interno di tag `point`. Si noti che, a differenza della campionatura, non è lecito l'uso dell'attributo `flags` né del tag `break`, e che le coordinate specificate sono tutte e sole quelle in corrispondenza delle quali si deve far scendere l'ago.

Il secondo caso, si assume che tutte le cuciture siano "a zig-zag" secondo la figura seguente:



Modello delle cuciture

La linea rossa rappresenta la posizione del filo, i cerchietti evidenziano i punti di cucitura e la linea blu evidenzia la "linea di base" della cucitura.

La descrizione della cucitura consta di 2 parti:

- in primo luogo vengono specificate le caratteristiche della cucitura direttamente nel tag `openStitchPath` o `closedStitchPath`:
  - la larghezza dello zig-zag viene specificato con l'attributo `width`;
  - la lunghezza del periodo della cucitura viene specificato con l'attributo `length`;

- trascurati il primo e l'ultimo semiperiodo, il parametro `swingPoints` specifica quanti punti di cucitura devono essere usati nel passare da un picco al successivo nella cucitura (1 in questo caso – i punti sui picchi non rientrano nel conteggio); il comportamento all'inizio e alla fine della cucitura è il seguente:
  - se `swingPoints` è dispari allora tra il primo punto e il primo picco devono essere apposti $\frac{\texttt{swingPoints}-1}{2}$ punti (è la soluzione più intuitiva, che consente di apporre punti equispaziati);
  - se `swingPoints` è pari allora l'implementazione è libera di apporre $\texttt{swingPoints}/2$ o $(\texttt{swingPoints}/2)-1$ punti.

  Deve essere obiettivo della macchina usare punti equispaziati, nel limite di quanto possa essere possibile ottenere con cuciture con linea di base non rettilinea;
- l'affrancatura all'inizio viene specificato indicando:
  - quante passate devono essere fatte (`beginBackTakingCount`);
  - quanti punti di cucitura deve essere lunga l'affrancatura (`beginBackTakingLength`).
- L'affrancatura alla fine è del tutto analoga a patto di usare i tag `endBackTakingCount` e `endBackTakingLength`;

• in secondo luogo si procede a descrivere la linea di base usando il meccanismo a tratti già visto per il file `2d.xml`, tolte le componenti semantiche che qui non hanno applicazione.

L'utilizzo del tag `break` è sconsigliato ma consentito. Quando è incontrato la macchina deve spostare l'ago dal punto precedente al successivo senza lasciare punti di cucitura intermedi, senza procedere alle affrancature (si considera di essere ancora all'interno della cucitura) e senza interrompere il filo.

È possibile specificare più cuciture diverse per lo stesso insieme di componenti (nelle medesime posizioni) specificando più tag `stitching` in successione.

Merita di osservare che un caso molto comune di cucitura, la cucitura "dritta", può essere descritta semplicemente impostando `width="0"`, `swingPoints="0"` e `length` pari alla lunghezza del passo desiderata.

Geometrie tridimensionali (`3d.xml`)

Le informazioni nel dominio tridimensionale sono relative a:

- geometrie della forma, che devono essere definibili sia al livello di dettaglio richiesto dalle operazioni di tornitura sia a quello richiesto per le operazioni di progettazione della scarpa (ivi compresa la separazione delle 3 superfici);
- definizione delle linee di lavorazione, che consentano il posizionamento automatico degli utensili;
- definizione delle caratteristiche essenziali di sottopiede di montaggio e di tacco, così da consentire l'automatizzazione delle operazioni di inchiodatura del tacco.

La rappresentazione dei dati proposta mira ad essere adeguata a questi scopi. Alcuni aspetti supplementari, come la possibilità di verificare la corrispondenza tra gli sviluppi fatti nel dominio 2d e quelli fatti nel dominio 3d, sono ancora in sviluppo e probabilmente entreranno a far parte dello standard in una prossima versione.

Formato del file

Il formato del file proposto è il seguente:

```
<root3d>
  <geometry id="37a">
    <description field="name" value="Forma bella"/>

    <coordinatesTransformation reference="anchorage" alpha="0" beta="30"
                                gamma="-10" offsetX="" offsetY="" offsetZ=""/>
    {... altre trasformazioni per altri punti di interesse ...}

    <sample id="flat" error="0.007">
      <ring id="1">
        <surfacePoint id="1" x="0.1" y="0.2" z="1" surface="top/upper/bottom"/>
        {... altri punti ...}
      </ring>
      {... altri anelli, con punti con id omologo ai precedenti ...}
    </sample>
    {... altri campionamenti ...}

    <work id="1" action="card/nail/glue">
      <workPath id="1" label="fore/back/left/right/bottom/side">
        <allowedMaterialsCombinations>
          <materialsCombination id="102"/>
          {... altre combinazioni di materiali così lavorabili ...}
        </allowedMaterialsCombinations>
        <workPoint x="0" y="0.8" z="0.01" theta="0" phi="0" size="10"/>
        {... altri punti ...}
      </workPath>
      {... altri percorsi di lavorazione ...}
    </work>
    {... altre lavorazioni ...}

    <heel>
      <coordinatesTransformation reference="heel" alpha="0" beta="30"
                              gamma="-10" offsetX="10" offsetY="-20" offsetZ="2"/>
      <heelFastening>
```

```
        <fastenMode name="glue" value="yes/no"/>
        <fastenMode name="nail" value="yes/no"/>
      </heelFastening>
      <insoleNailArea>
        <insoleArea id="0" nailsAllowed="yes/no">
          <basePoint x="0" y="0"/>
          {... altri punti ...}
        </insoleArea>
        {... altre aree ...}
      </insoleNailArea>
      <heelGeometry id="0">
        <heelArea id="0" nailsAllowed="yes/no">
          <basePoint x="0" y="0"/>
          {... altri punti ...}
        </heelArea>
        {... altre aree ...}
        <volume>
          <cone height="0.03">
            <vertex id="0" x="0" y="0" z="1"/>
            {... vertici delle altre aree ...}
          </cone>
          {... altri coni ...}
        </volume>
        <nailProfile id="0">
          <nail x="1" y="3" z="3" theta="0" phi="10" length="12"/>
          {... altri chiodi per questo profilo ...}
        </nailProfile>
        {... altri profili di inchiodatura utilizzabili ...}
      </heelGeometry>
    </heel>
  </geometry>

  <info id="Informazioni aggiuntive">
    {... informazioni aggiuntive ...}
  </info>
</root3d>
```

Tutte le informazioni sono racchiuse tra i tag `<root3d>` e `</root3d>`, e sono raggruppate per taglie (`geometry`). L'identificativo (`id`) della geometria identifica univocamente la taglia della calzatura, e deve coincidere con l'omonimo campo memorizzato nel file `2d.xml` per le informazioni bidimensionali relative alla medesima taglia.

Per ciascuna geometria è possibile specificare una o più matrici di trasformazione delle coordinate. La parametrizzazione scelta, che consente solo rototraslazioni, consiste nell'applicazione in sequenza di:

- una rotazione sul piano *xy* attorno all'origine di un angolo α (attributo `alpha`);
- una rotazione sul piano *xz* attorno all'origine di un angolo β (attributo `beta`);
- una rotazione sul piano *yz* attorno all'origine di un angolo γ (attributo `gamma`);
- una traslazione di un vettore $(\delta_x, \delta_y, \delta_z)$ (attributi `offsetX`, `offsetY` e `offsetZ`).

Queste operazioni, applicate nell'ordine, danno origine alla seguente matrice di trasformazione delle coordinate:

$$\begin{pmatrix} \cos\alpha\cos\beta & -\sin\alpha\cos\beta & -\sin\beta & \delta_x \\ \sin\alpha\cos\gamma-\cos\alpha\sin\beta\sin\gamma & \cos\alpha\cos\gamma+\sin\alpha\sin\beta\sin\gamma & -\cos\beta\sin\gamma & \delta_y \\ \sin\alpha\sin\gamma+\cos\alpha\sin\beta\cos\gamma & \cos\alpha\sin\gamma-\sin\alpha\sin\beta\cos\gamma & \cos\beta\cos\gamma & \delta_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Matrice di trasformazione delle coordinate

Analogamente a quanto descritto per il file `assembly.xml`, una trasformazione *T* trasforma le coordinate di un generico punto nello spazio in cui è descritto (x, y, z) nelle coordinate "di arrivo" $(\hat{x}, \hat{y}, \hat{z})$ secondo l'identità

$$\begin{pmatrix} \hat{x} \\ \hat{y} \\ \hat{z} \\ 1 \end{pmatrix} = T \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

L'attributo `reference` specifica lo scopo del sistema di riferimento indotto, e dovrà essere scelto in un insieme (ancora da definire) di valori possibili. Ad esempio, il valore `anchorage` potrà indicare che nel nuovo sistema di riferimento l'origine è posta nel "fulcro" (da meglio definire) di un afferraggio fissato rigidamente alla forma, in modo che questa giaccia nel semispazio a coordinata *z* negativa e abbia la punta rivolta verso valori positivi della *x*.

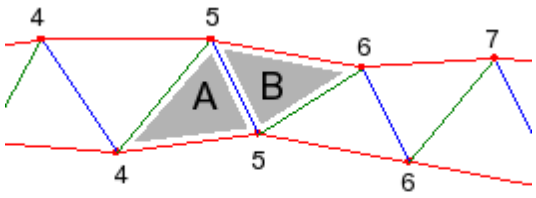
Geometria della forma

La geometria della forma è definita da un campionamento. A questo scopo occorre precisare che:

- la forma è posizionata in modo che l'asse *z* coincida con l'asse di tornitura, con la punta della scarpa rivolta verso valori positivi di tale coordinata;
- la descrizione avviene definendo un certo numero di linee chiuse attorno alla forma. Queste linee dovrebbero in linea di principio essere planari a coordinata *z* costante, tuttavia questo non è un vincolo in quanto riteniamo che avrebbe conseguenze marcate al momento del grading. Si raccomanda comunque di generare tali linee tenendo conto di questo aspetto;
- le linee di cui al punto precedente sono definite tramite un campionamento "rado" (indicativamente, adeguato per i progettisti di scarpe e per i produttori di macchine di montaggio e preparazione al montaggio, quali montafianchi/boette, cardatrici, incollatrici, ...). Il campionamento contiene sempre lo stesso numero di punti per ciascuna linea, e ciascun punto del campionamento possiede un identificatore che

consente di metterlo in relazione con un punto ("omologo") di un'altra linea. Tutti i punti appartengono alla superficie della forma;

- la mesh definita è a facce triangolari (si faccia riferimento alla figura seguente per maggiore chiarezza). I lati dei triangoli sono i segmenti che uniscono:
  - due punti successivi della stessa linea (tratti rossi);
  - punti di due linee adiacenti col medesimo identificatore (tratti blu);
  - punti di due linee adiacenti con identificatore consecutivo, a patto che l'identificatore del punto che giace sulla curva definita prima segua quello del punto definito sulla linea definita per seconda (tratti verdi).



Mesh triangolare

Assodato quanto sopra, la definizione della forma non dovrebbe apparire complessa. All'interno del tag `<sample>` vengono definiti più anelli (tag `<ring>`), ciascuno identificato dall'attributo `id`, all'interno dei quali sono specificati i punti del campionamento. Come si è detto, ogni identificativo usato all'interno di un anello deve essere usato anche in tutti gli altri (è necessario perché la mesh indicata sia ben definita), condizione che ha l'effetto collaterale di imporre che il numero di punti sia il medesimo per ciascun anello. Al tag `<sample>` è anche associata una misura di errore (`error`), che rappresenta il massimo scostamento della superficie campionata da quella "ideale" con un'interpolazione a triangoli piani.

Oltre all'`id` e alle coordinate, i punti sono caratterizzati da un attributo (`surface`) che specifica a quale delle 3 superfici (top, upper o bottom) appartenga il punto.

Per soddisfare le diverse necessità di attori diversi, è possibile fornire più campionamenti per la medesima forma. È richiesto che tutti i campionamenti forniti rappresentino la forma nello stesso sistema di riferimento.

Lavorazioni lineari

Le caratteristiche geometriche di molte lavorazioni possono essere definite tramite linee unidimensionali nello spazio tridimensionale. Rientrano in questa categoria ad esempio le premontatrici e le montafianchi/boette (posizione iniziale delle pinze, posizione finale, percorso

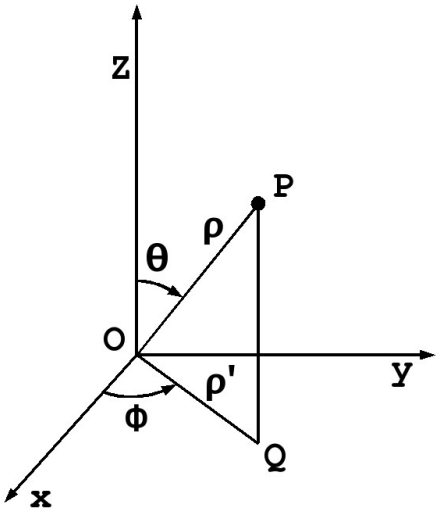
di incollaggio/inchiodatura), ribattitrici e, parzialmente, le macchine per l'iniezione diretta della suola su tomaia (per quanto il formato dati non consenta la descrizione completa dello stampo, ci sono dati sufficienti per la specificazione della linea di chiusura stampo). Anche la definizione precisa del filoforma può essere affrontata in questo modo.

Le linee di lavorazione relative ad operazioni affini (come ad esempio il montaggio della punta e dei fianchi) sono raggruppate con un tag `work`, che consente di specificare a quale lavorazione corrispondano i percorsi associati (tag `action`) e che è identificabile tra gli altri grazie ad un identificatore (`id`) univoco.

Ogni lavorazione è descritta da uno o più percorsi di lavorazione (`workPath`), ciascuno caratterizzato da un'identificatore (`id`, univoco all'interno del tag `work`), da un'etichetta (`label`, che consente di specificare la lavorazione alla quale appartiene il percorso – ad esempio, se un percorso di incollaggio sia relativo a una montafianchi o a una premonta) e da un riferimento alle combinazioni di materiali per la quale la lavorazione è valida.

L'indicazione delle combinazioni di materiali, per la quale si rimanda anche alla descrizione del file `materials.xml`, consente di gestire le lavorazioni sensibili al cambiamento del materiale (come la cardatura). L'idea è quella di specificare una sola volta le lavorazioni relative a combinazioni di materiali "simili" (dal punto di vista della lavorazione), separandole da combinazioni sensibilmente diverse (ad esempio per il differente spessore dei pellami) che richiedono altre linee di lavorazione.

La definizione del percorso vera e propria è fatta tramite una campionatura. Il formato dati consente di specificarla tramite una successione di punti (tag `<workPoint>`), caratterizzati



Coordinate sferiche

dalle 3 coordinate (`x`, `y`, `z`), da un orientamento (`theta` e `phi`, definite nella figura) e da una dimensione (`size`) la cui interpretazione dipende dalla specifica lavorazione.

Nella seguente tabella sono riportate le lavorazioni lineari attualmente previste, le `label` associate e l'interpretazione della direzione dei punti.

Lavorazione	***action***	***label***	***Interpretazione direzione***
Cardatura	`card`	`bottom` (fondo) `side` (fianco)	Nel caso della cardatura, il percorso fornito deve coincidere con il margine esterno, o con la parte superiore, dell'area da cardare, per il fondo e per il fianco rispettivamente. Il vettore direzione deve essere approssimativamente tangente al pellame, ortogonale al percorso e diretto verso l'area da lavorare. L'attributo `size` deve essere pari alla larghezza della lavorazione in ogni punto.
Montaggio	`nail`	`fore` (punta) `heel` (tallone) `left` (fianco sx) `right` (fianco dx)	Il percorso di lavorazione deve coincidere con la linea da inchiodare, la direzione con quella del chiodo (il verso della direzione deve coincidere col verso di inserimento dei chiodi) e l'attributo `size` con la lunghezza desiderata del chiodo.
	`glue`	`fore` (punta) `heel` (tallone) `left` (fianco sx) `right` (fianco dx)	Il percorso di lavorazione deve coincidere con la linea da incollare e la direzione con quella dei becchi per la colla (ove presenti). L'attributo `size` non ha significato e deve essere posto a 0.

Tacchi

Per sua natura, il processo di fissaggio dei tacchi non è assimilabile a quelli affrontati nella sezione precedente. Per questo anche la struttura dati è diversa, e definita a parte.

Sono definite le caratteristiche essenziali del sottopiede, supposto planare per essere più gestibile, e le caratteristiche significative del tacco, anch'esso con base planare per semplicità (si assume che la base del tacco sia spianata "riempiendone" la concavità, e che il sottopiede sia corrispondentemente "scavato").

Le informazioni necessarie per il fissaggio del tacco sono raggruppate nel tag `<heel>`. Nell'ordine, sono riportate:

- una trasformazione che consente di collocare la soletta e il tacco nella corretta posizione rispetto alla forma;
- una descrizione (tag `<heelFastening>`) di quali metodologie utilizzare per il fissaggio del tacco. In particolare bisogna specificare se è necessaria o meno la colla (`glue`) e l'inchiodatura (`nail`)
- una descrizione planare della soletta, che consente di definire quali aree sono attraversabili da chiodi;
- la geometria tridimensionale del tacco, semplificata e al solo scopo di definire quali siano le aree utili per ospitare un chiodo.

La descrizione della soletta (`<insoleNailArea>`) consta della definizione di un numero arbitrario aree chiuse (`<insoleArea>`), delle quali è fornito il perimetro tramite un campionamento. A tali aree è associato in identificatore (`id`), univoco all'interno della singola sezione `<insoleNailArea>`, e un flag `nailsAllowed` che specifica se in quell'area è lecito apporre chiodi. È consentito avere aree parzialmente o totalmente sovrapposte.

Si considera che l'area attraversabile dai chiodi sia quella composta da tutti e soli i punti che appartengono ad almeno un'area con `nailsAllowed="yes"` e non appartengono a nessuna area con `nailsAllowed="no"`.

La descrizione del tacco inizia in modo simile a quella della soletta. Il tacco è posizionato sopra questa (coordinata *z* positiva), ed è descritto nel medesimo sistema di riferimento.

Per prima cosa, come nel caso della soletta si identifica l'area inchiodabile della base del tacco (cioè la superficie del tacco che sarà a contatto col sottopiede), in modo analogo a quanto fatto per il sottopiede a patto di sostituire i tag `<insoleNailArea>` e `<insoleArea>` con `<heelGeometry>` e `<heelArea>` rispettivamente.

A differenza di quanto accade per il sottopiede, la descrizione del tacco include due descrizioni supplementari, costituite dai tag `<volume>` e `<nailProfile>`.

La sezione `<volume>` consente di specificare una approssimazione della struttura tridimensionale del tacco sufficiente per la determinazione automatica delle posizioni lecite per i chiodi. Tale struttura è specificata come segue:

- ad ogni `<heelArea>` è associato uno sviluppo tridimensionale a forma di tronco di cono (tag `<cone>`), che è specificato indicandone le coordinate del vertice (tag `<vertex>`, attributi `x`, `y` e `z`) e l'altezza (`height`) (il cono viene "tagliato" da un piano parallelo al piano *xy* a distanza `height` dal piano di base);
- la base superiore del tronco di cono così definito può essere iterativamente usata come nuova base per l'indicazione di un successivo tronco di cono, così da specificare una sovrapposizione di un numero arbitrariamente elevato di tronchi di cono;
- all'interno di ogni sezione `<cone>` deve essere specificato una e una sola volta il vertice per ciascuna delle aree di base definite (il tag `id` di `<vertex>` è un riferimento al tag `id` di `<nailArea>`);
- i volumi conservano la proprietà di poter essere occupati da un chiodo delle basi di partenza;
- il volume del tacco è definito come l'insieme di tutti e soli i punti che appartengono ad almeno un volume inchiodabile e a nessun volume non inchiodabile.

È possibile definire tronchi di cono che si "allargano" specificando un vertice "sotto" la base, cioè con coordinata *z* inferiore.

È infine possibile specificare un insieme di set di chiodi (tag `<nailProfile>`), che specificano un insieme di chiodi (tag `<nail>`, che racchiude le coordinate del punto di ingresso del chiodo, orientamento e lunghezza) che consentono il fissaggio adeguato del tacco. Qualora vengano specificati più set, la scelta di quello da utilizzare (ammesso che almeno uno sia ammissibile con la macchina utilizzata) è libera.

Informazioni aggiuntive

È possibile specificare informazioni aggiuntive, già formattate graficamente per essere stampate o visualizzate, utilizzando il tag opzionale `<info>`. Per le regole sull'uso di tale tag si vedano i dettagli relativi a `general.xml`

Specifica dei materiali (`materials.xml`)

In generale per ciascuna scarpa è possibile identificare:

- modello base
- variante
- riferimento materiali

Le prime due voci sono specificate in `general.xml`. Compito del file `materials.xml` è specificare tutte le combinazioni di materiali che sono state proposte per il modello/variante in oggetto.

Il file associa a ciascun identificatore di materiale (tag `materialClass` nel file `2d.xml`) a un materiale specifico. Ciascun materiale è descritto da in identificatore (`id`) e da una misura di spessore (`thickness`, usabile ad esempio per le operazioni di spaccatura) racchiusi nel tag `material`.

Le associazioni classe–materiale sono raggruppate in un tag `materialsCombination`, così che sia possibile averne più d'una (distinguibile dall'`id`) all'interno del singolo file.

È possibile specificare informazioni aggiuntive, già formattate graficamente per essere stampate o visualizzate, utilizzando il tag opzionale `info`. Per le regole sull'uso di tale tag si vedano i dettagli relativi a `general.xml`

La struttura del file risultante è la seguente:

```
<rootMaterials>
  <materialsCombination id="102">
    <description field="name" value="Coccodrillo verde"/>
    <materialDefinition materialClassId="23">
      <description field="name" value="Coccodrillo - fianco"/>
      <material id="10" thickness="1.5"/>
    </materialDefinition>
    {... altre definizioni di materiale ...}
  </materialsCombination>
  {... altre combinazioni di materiale ...}
  <info id="Informazioni aggiuntive">
    {... informazioni aggiuntive ...}
  </info>
</rootMaterials>
```

Ordine di produzione (`production.xml`)

Nell'ordine di produzione occorre specificare quante scarpe occorre produrre per taglia e per lato. Oltre all'ordine di scarpe intere, è possibile ordinare la produzione di singoli componenti supplementari – di cui va specificata la quantità – e anche specificare se alcune componenti delle scarpe intere vanno invece prodotte in numero minore.
È possibile specificare informazioni aggiuntive, già formattate graficamente per essere stampate o visualizzate, utilizzando il tag opzionale `info`, sia a livello di singolo ordine sia a livello globale. Per le regole sull'uso di tale tag si vedano i dettagli relativi a `general.xml`.

Il file risultante è così strutturato:

```
<rootProduction>
  <order id="1" materialsCombinationId="102">
    <quantity>
      <geometry id="3710" leftCount="3" rightCount="3">
        <additional componentId="2" side="right" count="1"/>
        {... altre componenti extra ...}
        <except componentId="2" side="right" count="1"/>
        {... altre componenti escluse ...}
      </geometry>
      {... quantità da produrre per altre taglie ...}
    </quantity>

    <info id="Note">
      {... note ...}
    </info>
  </order>
  {... altri ordini ...}
  <info id="Informazioni aggiuntive">
    {... informazioni aggiuntive ...}
  </info>
</rootProduction>
```

Capitolo 4 – Schemi XML

Quelli che seguono sono i documenti di struttura (XML Schema 1.0) per alcuni dei tipi di file definiti. Si tenga presente che sono ancora preliminari, e che pertanto non tutti i vincoli sul documento sono espressi.

general.xsd

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<!--
  XML Schema definition for the "general.xml" file defined by the Assomac
  JANE standard.

  This document conforms to the revision 0.7.0 of the standard document.
-->
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <!-- Main structure -->
  <xsd:element name="rootGeneral">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="description" type="descriptionType" minOccurs="0"
                                                               maxOccurs="unbounded"/>
        <xsd:element name="info" type="infoType" minOccurs="0"
                                                               maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>

  <!--
    description tags
  -->
  <xsd:complexType name="descriptionType">
    <xsd:attribute name="field" type="xsd:string" use="required"/>
    <xsd:attribute name="value" type="xsd:string" use="required"/>
  </xsd:complexType>

  <!--
    extended information tag
  -->
  <xsd:complexType name="infoType">
    <xsd:simpleContent>
      <xsd:extension base="xsd:string">
        <xsd:attribute name="id" type="xsd:string" use="required"/>
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>
</xsd:schema>
```

2d.xsd

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<!--
  XML Schema definition for the "2d.xml" file defined by the Assomac
  JANE standard.

  This document conforms to the revision 0.7.0 of the standard document.
-->
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
```

```
<xsd:element name="root2d">
  <!--
    File global structure, compound by:
     - a collection of component-dependent specifications;
     - a collection of geometric-dependent specifications;
     - an optional collection of supplemental informations.
  -->
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="componentDependent" type="componentDependentType"
                                          minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element name="geometryDependent" type="geometryDependentType"
                                           minOccurs="0" maxOccurs="unbounded">
        <!-- INTERNAL KEY CONSTRAINTS -->
        <xsd:key name="constraint-internal-4">
          <xsd:selector xpath="component"/>
          <xsd:field xpath="@id"/>
        </xsd:key>
      </xsd:element>

      <xsd:element name="info" minOccurs="0" maxOccurs="unbounded">
        <xsd:complexType>
          <xsd:simpleContent>
            <xsd:extension base="xsd:string">
              <xsd:attribute name="id" type="xsd:string"/>
            </xsd:extension>
          </xsd:simpleContent>
        </xsd:complexType>
      </xsd:element>
    </xsd:sequence>
  </xsd:complexType>

  <!-- INTERNAL KEY CONSTRAINTS -->
  <xsd:key name="constraint-internal-1">
    <xsd:selector xpath="componentDependent"/>
    <xsd:field xpath="@id"/>
  </xsd:key>
  <!-- INTERNAL KEY CONSTRAINTS -->
  <xsd:key name="constraint-internal-3">
    <xsd:selector xpath="geometryDependent"/>
    <xsd:field xpath="@id"/>
  </xsd:key>
  <!-- INTERNAL KEY CONSTRAINTS -->
  <xsd:key name="constraint-internal-6">
    <xsd:selector xpath="info"/>
    <xsd:field xpath="@id"/>
  </xsd:key>

  <!-- EXTERNAL KEY CONSTRAINTS -->
  <xsd:keyref name="constraint-external-1" refer="constraint-internal-4">
    <xsd:selector xpath="componentDependent"/>
    <xsd:field xpath="@id"/>
  </xsd:keyref>
  <!-- EXTERNAL KEY CONSTRAINTS -->
  <xsd:keyref name="constraint-external-2" refer="constraint-internal-1">
    <xsd:selector xpath="geometryDependent/component"/>
    <xsd:field xpath="@id"/>
  </xsd:keyref>
  <!-- EXTERNAL KEY CONSTRAINTS -->
  <xsd:keyref name="constraint-external-3" refer="constraint-internal-5b">
    <xsd:selector xpath="componentDependent/slicing/tract"/>
    <xsd:field xpath="@id"/>
  </xsd:keyref>
  <!-- EXTERNAL KEY CONSTRAINTS -->
  <xsd:keyref name="constraint-external-4" refer="constraint-internal-3">
    <xsd:selector xpath="geometryDependent/component/sameAsGeometry"/>
    <xsd:field xpath="@id"/>
  </xsd:keyref>
  <!-- EXTERNAL KEY CONSTRAINTS -->
```

```
  <xsd:keyref name="constraint-external-5" refer="constraint-internal-5b">
    <xsd:selector xpath="geometryDependent/component/perimeter/tract"/>
    <xsd:field xpath="@id"/>
  </xsd:keyref>
</xsd:element>

<!--
  componentDependent
-->
<xsd:complexType name="componentDependentType">
  <xsd:sequence>
    <xsd:element name="description" type="descriptionType" minOccurs="0"
                                                          maxOccurs="unbounded"/>
    <xsd:element name="slicing" type="slicingType" minOccurs="0"
                                                           maxOccurs="unbounded">
      <!-- INTERNAL KEY CONSTRAINTS -->
      <xsd:key name="constraint-internal-2">
        <xsd:selector xpath="tract"/>
        <xsd:field xpath="@id"/>
      </xsd:key>
    </xsd:element>
  </xsd:sequence>
  <xsd:attribute name="id" type="xsd:string" use="required"/>
  <xsd:attribute name="materialClassId" type="xsd:string" use="required"/>
  <xsd:attribute name="leftCount" type="xsd:nonNegativeInteger"
                                                             use="required"/>
  <xsd:attribute name="rightCount" type="xsd:nonNegativeInteger"
                                                             use="required"/>
  <xsd:attribute name="side" type="leftRight" use="required"/>
  <xsd:attribute name="direction" type="xsd:float" use="required"/>
  <xsd:attribute name="directionTolerance" type="xsd:float" use="required"/>
</xsd:complexType>

<!--
  geometryDependent
-->
<xsd:complexType name="geometryDependentType">
  <xsd:sequence>
    <xsd:element name="component" minOccurs="0" maxOccurs="unbounded">
      <xsd:complexType>
        <!--
          Components can be defined in two ways:
           - by giving their properties and geometric description;
           - by referencing the same component for another geometry.
        -->
        <xsd:choice>
          <!-- Define properties and geometry -->
          <xsd:sequence>
            <xsd:group ref="perimeterWorkPath"/>
            <xsd:choice minOccurs="0" maxOccurs="unbounded">
              <xsd:group ref="closedWorkPath"/>
              <xsd:group ref="openWorkPath"/>
              <xsd:group ref="areaPath"/>
            </xsd:choice>
          </xsd:sequence>

          <!-- Use properties and description from another geometry -->
          <xsd:element name="sameAsGeometry">
            <xsd:complexType>
              <xsd:attribute name="id" type="xsd:string" use="required"/>
            </xsd:complexType>
          </xsd:element>
        </xsd:choice>
        <xsd:attribute name="id" type="xsd:string" use="required"/>
      </xsd:complexType>
    </xsd:element>
  </xsd:sequence>
  <xsd:attribute name="id" type="xsd:string" use="required"/>
  <xsd:attribute name="sizeUnit" type="sizeUnit" use="required"/>
```

```
  <xsd:attribute name="sizeValue" type="nonNegativeFloat" use="required"/>
  <xsd:attribute name="footLength" type="nonNegativeFloat" use="required"/>
  <xsd:attribute name="toeWidth" type="nonNegativeFloat" use="optional"/>
  <xsd:attribute name="foreFit" type="nonNegativeFloat" use="optional"/>
  <xsd:attribute name="heelFit" type="nonNegativeFloat" use="optional"/>
</xsd:complexType>

<!--
  Slicing & folding type
-->
<xsd:complexType name="slicingType">
  <xsd:sequence minOccurs="1" maxOccurs="unbounded">
    <xsd:element name="slicedTract">
      <xsd:complexType>
        <xsd:attribute name="id" type="xsd:string" use="required"/>
      </xsd:complexType>
    </xsd:element>
  </xsd:sequence>
  <xsd:attribute name="depth" type="nonNegativeFloat" use="required"/>
  <xsd:attribute name="lift" type="nonNegativeFloat" use="required"/>
  <xsd:attribute name="angle" type="nonNegativeFloat" use="required"/>
  <xsd:attribute name="visibleSide" type="yesNo" use="required"/>
  <xsd:attribute name="folding" type="foldingType" use="required"/>
</xsd:complexType>

<!--
  General structure of a closed work path
-->
<xsd:group name="closedWorkPath">
  <xsd:sequence>
    <xsd:element name="closedPath">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:group ref="closedPathDefinitionType"/>
        </xsd:sequence>
        <xsd:attribute name="id" type="xsd:string" use="required"/>
        <xsd:attribute name="action" type="actionWorkType" use="required"/>
        <xsd:attribute name="mechanicalQuality" type="highLow"
                                                          use="optional"/>
        <xsd:attribute name="aestheticQuality" type="highLow" use="optional"/>
        <xsd:attribute name="toolSide" use="optional" default="0">
          <xsd:simpleType>
            <xsd:restriction base="xsd:float">
              <xsd:minInclusive value="-1"/>
              <xsd:maxInclusive value="1"/>
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:attribute>
      </xsd:complexType>

      <!-- INTERNAL KEY CONSTRAINTS -->
      <xsd:key name="constraint-internal-5a">
        <xsd:selector xpath="tract"/>
        <xsd:field xpath="@id"/>
      </xsd:key>
    </xsd:element>
  </xsd:sequence>
</xsd:group>

<!--
  General structure of a component perimeter
-->
<xsd:group name="perimeterWorkPath">
  <xsd:sequence>
    <xsd:element name="perimeter">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:group ref="closedPathDefinitionType"/>
        </xsd:sequence>
```

```
        <xsd:attribute name="id" type="xsd:string" use="required"/>
        <xsd:attribute name="action" type="actionWorkType" use="required"
                                                              fixed="cut"/>
        <xsd:attribute name="toolSide" use="optional" default="0">
          <xsd:simpleType>
            <xsd:restriction base="xsd:float">
              <xsd:minInclusive value="-1"/>
              <xsd:maxInclusive value="1"/>
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:attribute>
      </xsd:complexType>

      <!-- INTERNAL KEY CONSTRAINTS -->
      <xsd:key name="constraint-internal-5b">
        <xsd:selector xpath="tract"/>
        <xsd:field xpath="@id"/>
      </xsd:key>
    </xsd:element>
  </xsd:sequence>
</xsd:group>

<!--
  General structure of an open work path
-->
<xsd:group name="openWorkPath">
  <xsd:sequence>
    <xsd:element name="openPath">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:group ref="openPathDefinitionType"/>
        </xsd:sequence>
        <xsd:attribute name="id" type="xsd:string" use="required"/>
        <xsd:attribute name="action" type="actionWorkType" use="required"/>
        <xsd:attribute name="mechanicalQuality" type="highLow"
                                                          use="optional"/>
        <xsd:attribute name="aestheticQuality" type="highLow" use="optional"/>
        <xsd:attribute name="toolSide" use="optional" default="0">
          <xsd:simpleType>
            <xsd:restriction base="xsd:float">
              <xsd:minInclusive value="-1"/>
              <xsd:maxInclusive value="1"/>
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:attribute>
      </xsd:complexType>

      <!-- INTERNAL KEY CONSTRAINTS -->
      <xsd:key name="constraint-internal-5c">
        <xsd:selector xpath="tract"/>
        <xsd:field xpath="@id"/>
      </xsd:key>
    </xsd:element>
  </xsd:sequence>
</xsd:group>

<!--
  General structure of an quality area definition path
-->
<xsd:group name="areaPath">
  <xsd:sequence>
    <xsd:element name="area">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:group ref="closedPathDefinitionType"/>
        </xsd:sequence>
        <xsd:attribute name="id" type="xsd:string" use="required"/>
        <xsd:attribute name="action" type="actionWorkType" use="required"/>
        <xsd:attribute name="mechanicalQuality" type="highLow"
```

```
                                                                    use="optional"/>
        <xsd:attribute name="aestheticQuality" type="highLow" use="optional"/>
      </xsd:complexType>

      <!-- INTERNAL KEY CONSTRAINTS -->
      <xsd:key name="constraint-internal-5d">
        <xsd:selector xpath="tract"/>
        <xsd:field xpath="@id"/>
      </xsd:key>
    </xsd:element>
  </xsd:sequence>
</xsd:group>

<!--
  Actions
-->
<xsd:simpleType name="actionWorkType">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="cut"/>
    <xsd:enumeration value="etch"/>
    <xsd:enumeration value="mark"/>
    <xsd:enumeration value="show"/>
    <xsd:enumeration value="quality"/>
  </xsd:restriction>
</xsd:simpleType>

<!--
  Geometric definition for a closed path
-->
<xsd:group name="closedPathDefinitionType">
  <xsd:sequence>
    <xsd:group ref="pointTractGroup" maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:group>

<!--
  point-tract group
-->
<xsd:group name="pointTractGroup">
  <xsd:sequence>
    <xsd:element name="point" type="pointType"/>
    <xsd:element name="tract" type="tractType"/>
  </xsd:sequence>
</xsd:group>

<!--
  Geometric definition for an open path
-->
<xsd:group name="openPathDefinitionType">
  <xsd:sequence>
    <xsd:element name="point" type="pointType"/>
    <xsd:group ref="tractPointGroup" maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:group>

<!--
  tract-point group
-->
<xsd:group name="tractPointGroup">
  <xsd:sequence>
    <xsd:element name="tract" type="tractType"/>
    <xsd:element name="point" type="pointType"/>
  </xsd:sequence>
</xsd:group>

<!--
  Generic tract
-->
<xsd:complexType name="tractType">
```

```
  <xsd:sequence>
    <xsd:choice>
      <!--
        There are two possibile situations:
         - if one needs to define a text, a semanticText definition
           is sufficient, sampled and vector definitions are optional
           and notch definitions are forbidden;
         - if one doesn't want to describe text, the sampled definition
           is mandatory, a vector definition is optional and a single
           notch definition is also optional, while a semanticText
           definition is forbidden.
        The order of tags is the following:
         - semanticText
         - sampled
         - vector
         - referenceNotch/symbolicNotch
      -->
      <xsd:sequence>
        <xsd:element name="semanticText" type="semanticTextWayType"/>
        <xsd:element name="sampled" type="sampledWayType" minOccurs="0"/>
        <xsd:element name="vector" type="vectorWayType" minOccurs="0"/>
      </xsd:sequence>
      <xsd:sequence>
        <xsd:element name="sampled" type="sampledWayType"/>
        <xsd:element name="vector" type="vectorWayType" minOccurs="0"/>
        <xsd:choice minOccurs="0">
          <xsd:element name="referenceNotch" type="referenceNotchWayType"/>
          <xsd:element name="symbolicNotch" type="symbolicNotchWayType"/>
        </xsd:choice>
      </xsd:sequence>
    </xsd:choice>
  </xsd:sequence>
  <xsd:attribute name="id" type="xsd:string" use="required"/>
  <xsd:attribute name="smoothness" type="highLow" use="required"/>
  <xsd:attribute name="fidelity" type="highLow" use="required"/>
  <xsd:attribute name="maxToolSize" type="positiveFloat" use="optional"
                                                              default="INF"/>
</xsd:complexType>

<!--
  sampled tract
-->
<xsd:complexType name="sampledWayType">
  <xsd:sequence>
    <xsd:element name="break" type="emptyType" minOccurs="0"/>
    <xsd:sequence minOccurs="0" maxOccurs="unbounded">
      <xsd:element name="point" type="pointType"/>
      <xsd:element name="break" type="emptyType" minOccurs="0"/>
    </xsd:sequence>
  </xsd:sequence>
</xsd:complexType>

<!--
  symbolicText tract
-->
<xsd:complexType name="semanticTextWayType">
  <xsd:attribute name="height" type="nonNegativeFloat" use="required"/>
  <xsd:attribute name="text" type="xsd:string" use="required"/>
</xsd:complexType>

<!--
  referenceNotch
-->
<xsd:complexType name="referenceNotchWayType">
  <xsd:attribute name="side" type="leftRight"/>
  <xsd:attribute name="depth" type="xsd:float"/>
  <xsd:attribute name="angle" type="xsd:float"/>
</xsd:complexType>
```

```
<!--
  symbolicNotch
-->
<xsd:complexType name="symbolicNotchWayType">
  <xsd:attribute name="side" type="leftRight"/>
  <xsd:attribute name="allowOpposite" type="yesNo"/>
</xsd:complexType>

<!--
  Base vector definition, compound by a periodicity of points and
  geometric directives
-->
<xsd:complexType name="vectorWayType">
  <xsd:sequence>
    <xsd:group ref="vectorEntity"/>
    <xsd:sequence minOccurs="0" maxOccurs="unbounded">
      <xsd:element name="point" type="pointType"/>
      <xsd:group ref="vectorEntity"/>
    </xsd:sequence>
  </xsd:sequence>
</xsd:complexType>

<!--
  Geometric entities
-->
<xsd:group name="vectorEntity">
  <xsd:choice>
    <xsd:element name="circle" type="circleType"/>
    <xsd:element name="arc" type="arcType"/>
    <xsd:element name="bezier" type="bezierType"/>
    <xsd:element name="break" type="breakType"/>
  </xsd:choice>
</xsd:group>

<!--
  Circle
-->
<xsd:complexType name="circleType"/>

<!--
  break
-->
<xsd:complexType name="breakType"/>

<!--
  Arc
-->
<xsd:complexType name="arcType">
  <xsd:attribute name="bulge" type="xsd:float" use="required"/>
</xsd:complexType>

<!--
  Bézier curve
-->
<xsd:complexType name="bezierType">
  <xsd:sequence minOccurs="0" maxOccurs="unbounded">
    <xsd:element name="bezierControlPoint" type="bezierControlPointType"/>
  </xsd:sequence>
</xsd:complexType>

<!--
  Empty data type
-->
<xsd:complexType name="emptyType"/>

<!--
  Point
-->
<xsd:complexType name="pointType">
```

```
  <xsd:attribute name="x" type="xsd:float" use="required"/>
  <xsd:attribute name="y" type="xsd:float" use="required"/>
  <xsd:attribute name="flags" type="pointFlagsType" use="optional"
                                                              default=""/>
</xsd:complexType>

<!--
    Bezier control point
-->
<xsd:complexType name="bezierControlPointType">
  <xsd:attribute name="x" type="xsd:float" use="required"/>
  <xsd:attribute name="y" type="xsd:float" use="required"/>
</xsd:complexType>

<!--
  Point flags
-->
<xsd:simpleType name="pointFlagsType">
  <xsd:restriction base="xsd:string">
    <xsd:pattern value="smooth|sharp|"/>
  </xsd:restriction>
</xsd:simpleType>

<!--
  Non-negative float data type
-->
<xsd:simpleType name="nonNegativeFloat">
  <xsd:restriction base="xsd:float">
    <xsd:minInclusive value="0"/>
  </xsd:restriction>
</xsd:simpleType>

<!--
  Positive float data type
-->
<xsd:simpleType name="positiveFloat">
  <xsd:restriction base="xsd:float">
    <xsd:minExclusive value="0"/>
  </xsd:restriction>
</xsd:simpleType>

<!--
  Allowed shoe size units
-->
<xsd:simpleType name="sizeUnit">
  <xsd:restriction base="xsd:string">
    <xsd:pattern
        value="au-[mw]|en|eu|jp-[mw]|mp|mx|us-[mwbg]|uk-[mwbg]"/>
  </xsd:restriction>
</xsd:simpleType>

<!--
  description tags
-->
<xsd:complexType name="descriptionType">
  <xsd:attribute name="field" type="xsd:string" use="required"/>
  <xsd:attribute name="value" type="xsd:string" use="required"/>
</xsd:complexType>

<!--
  left/right data type
-->
<xsd:simpleType name="leftRight">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="left"/>
    <xsd:enumeration value="right"/>
  </xsd:restriction>
</xsd:simpleType>
```

```
  <!--
      high/low data type
      -->
  <xsd:simpleType name="highLow">
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="high"/>
      <xsd:enumeration value="low"/>
    </xsd:restriction>
  </xsd:simpleType>

  <!--
    yes/no data type
  -->
  <xsd:simpleType name="yesNo">
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="yes"/>
      <xsd:enumeration value="no"/>
    </xsd:restriction>
  </xsd:simpleType>

  <!--
    none/simple/reinforces data type (for folding)
  -->
  <xsd:simpleType name="foldingType">
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="none"/>
      <xsd:enumeration value="simple"/>
      <xsd:enumeration value="reinforced"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:schema>
```

assembly.xsd

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<!--
  XML Schema definition for the "assembly.xml" file defined by the Assomac
  JANE standard.

  This document conforms to the revision 0.7.0 of the standard document.
-->
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="rootAssembly">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="geometry" type="geometryType"
                                          minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:complexType>

    <!--
      Uniqueness constrains
    -->
    <!-- geometry id -->
    <xsd:unique name="dummy1">
      <xsd:selector xpath="rootAssembly"/>
      <xsd:field xpath="geometry/@id"/>
    </xsd:unique>
  </xsd:element>

  <!--
    geometryType
  -->
  <xsd:complexType name="geometryType">
    <xsd:sequence>
      <xsd:element name="stitching">
        <xsd:complexType>
```

```
        <xsd:sequence>
          <xsd:element name="description" type="descriptionType" minOccurs="0"
                                                    maxOccurs="unbounded"/>
          <xsd:element name="components">
            <xsd:complexType>
              <xsd:sequence minOccurs="1" maxOccurs="unbounded">
                <xsd:element name="component">
                  <xsd:complexType>
                    <xsd:attribute name="id" type="xsd:string"/>
                    <xsd:attribute name="flipX" type="yesNo"/>
                    <xsd:attribute name="rotation" type="xsd:float"/>
                    <xsd:attribute name="offsetX" type="xsd:float"/>
                    <xsd:attribute name="offsetY" type="xsd:float"/>
                  </xsd:complexType>
                </xsd:element>
              </xsd:sequence>
            </xsd:complexType>
          </xsd:element>
          <xsd:element name="path">
            <xsd:complexType>
              <xsd:choice>
                <xsd:sequence>
                  <xsd:element name="strictSample" type="strictSampledType"/>
                  <xsd:choice minOccurs="0">
                    <xsd:group ref="closedStitchPathType"/>
                    <xsd:group ref="openStitchPathType"/>
                  </xsd:choice>
                </xsd:sequence>
                <xsd:sequence>
                  <xsd:choice>
                    <xsd:group ref="closedStitchPathType"/>
                    <xsd:group ref="openStitchPathType"/>
                  </xsd:choice>
                </xsd:sequence>
              </xsd:choice>
              <xsd:attribute name="id" type="xsd:string"/>
            </xsd:complexType>
          </xsd:element>
        </xsd:sequence>
        <xsd:attribute name="id" type="xsd:string"/>
      </xsd:complexType>
    </xsd:element>
  </xsd:sequence>
  <xsd:attribute name="id" type="xsd:string" use="required"/>
</xsd:complexType>

<!--
  General structure of a closed work path
-->
<xsd:group name="closedStitchPathType">
  <xsd:sequence>
    <xsd:element name="closedStitchPath">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:group ref="closedPathDefinitionType"/>
        </xsd:sequence>
        <xsd:attribute name="width" type="nonNegativeFloat"/>
        <xsd:attribute name="length" type="positiveFloat"/>
        <xsd:attribute name="swingPoints" type="positiveInteger"/>
        <xsd:attribute name="beginBackTackingCount" type="positiveInteger"/>
        <xsd:attribute name="endBackTackingCount" type="positiveInteger"/>
        <xsd:attribute name="beginBackTackingLength" type="positiveInteger"/>
        <xsd:attribute name="endBackTackingLength" type="positiveInteger"/>
      </xsd:complexType>
    </xsd:element>
  </xsd:sequence>
</xsd:group>

<!--
```

```
  General structure of an open work path
-->
<xsd:group name="openStitchPathType">
  <xsd:sequence>
    <xsd:element name="openStitchPath">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:group ref="openPathDefinitionType"/>
        </xsd:sequence>
        <xsd:attribute name="width" type="nonNegativeFloat"/>
        <xsd:attribute name="length" type="positiveFloat"/>
        <xsd:attribute name="swingPoints" type="xsd:nonNegativeInteger"/>
        <xsd:attribute name="beginBackTackingCount" type="positiveInteger"/>
        <xsd:attribute name="endBackTackingCount" type="positiveInteger"/>
        <xsd:attribute name="beginBackTackingLength" type="positiveInteger"/>
        <xsd:attribute name="endBackTackingLength" type="positiveInteger"/>
      </xsd:complexType>
    </xsd:element>
  </xsd:sequence>
</xsd:group>

<!--
  Geometric definition for a closed path
-->
<xsd:group name="closedPathDefinitionType">
  <xsd:sequence>
    <xsd:group ref="pointTractGroup" maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:group>

<!--
  point-tract group
-->
<xsd:group name="pointTractGroup">
  <xsd:sequence>
    <xsd:element name="point" type="pointType"/>
    <xsd:element name="tract" type="tractType"/>
  </xsd:sequence>
</xsd:group>

<!--
  Geometric definition for an open path
-->
<xsd:group name="openPathDefinitionType">
  <xsd:sequence>
    <xsd:element name="point" type="pointType"/>
    <xsd:group ref="tractPointGroup" maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:group>

<!--
  tract-point group
-->
<xsd:group name="tractPointGroup">
  <xsd:sequence>
    <xsd:element name="tract" type="tractType"/>
    <xsd:element name="point" type="pointType"/>
  </xsd:sequence>
</xsd:group>

<!--
  Generic tract
-->
<xsd:complexType name="tractType">
  <xsd:sequence>
    <xsd:element name="sampled" type="sampledWayType"/>
    <xsd:element name="vector" type="vectorWayType" minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute name="id" type="xsd:string" use="required"/>
```

```
</xsd:complexType>

<!--
  sampled tract
-->
<xsd:complexType name="sampledWayType">
  <xsd:sequence>
    <xsd:element name="break" type="emptyType" minOccurs="0"/>
    <xsd:sequence minOccurs="0" maxOccurs="unbounded">
      <xsd:element name="point" type="pointType"/>
      <xsd:element name="break" type="emptyType" minOccurs="0"/>
    </xsd:sequence>
  </xsd:sequence>
</xsd:complexType>

<!--
  strict sampled tract
-->
<xsd:complexType name="strictSampledType">
  <xsd:sequence minOccurs="0" maxOccurs="unbounded">
    <xsd:element name="point" type="pointType"/>
  </xsd:sequence>
</xsd:complexType>

<!--
  Base vector definition, compound by a periodicity of points and
  geometric directives
-->
<xsd:complexType name="vectorWayType">
  <xsd:sequence>
    <xsd:group ref="vectorEntity"/>
    <xsd:sequence minOccurs="0" maxOccurs="unbounded">
      <xsd:element name="point" type="pointType"/>
      <xsd:group ref="vectorEntity"/>
    </xsd:sequence>
  </xsd:sequence>
</xsd:complexType>

<!--
  Geometric entities
-->
<xsd:group name="vectorEntity">
  <xsd:choice>
    <xsd:element name="circle" type="circleType"/>
    <xsd:element name="arc" type="arcType"/>
    <xsd:element name="bezier" type="bezierType"/>
    <xsd:element name="break" type="breakType"/>
  </xsd:choice>
</xsd:group>

<!--
  Circle
-->
<xsd:complexType name="circleType"/>

<!--
  break
-->
<xsd:complexType name="breakType"/>

<!--
  Arc
-->
<xsd:complexType name="arcType">
  <xsd:attribute name="bulge" type="xsd:float" use="required"/>
</xsd:complexType>

<!--
  Bézier curve
```

```
-->
<xsd:complexType name="bezierType">
  <xsd:sequence minOccurs="0" maxOccurs="unbounded">
    <xsd:element name="point" type="pointType"/>
  </xsd:sequence>
</xsd:complexType>

<!--
  Empty data type
-->
<xsd:complexType name="emptyType"/>

<!--
  Point
-->
<xsd:complexType name="pointType">
  <xsd:attribute name="x" type="xsd:float" use="required"/>
  <xsd:attribute name="y" type="xsd:float" use="required"/>
  <xsd:attribute name="flags" type="pointFlagsType" use="optional"
                                                                  default=""/>
</xsd:complexType>

<!--
  Point flags
-->
<xsd:simpleType name="pointFlagsType">
  <xsd:restriction base="xsd:string">
    <xsd:pattern value="smooth|sharp|"/>
  </xsd:restriction>
</xsd:simpleType>

<!--
  Non-negative float data type
-->
<xsd:simpleType name="nonNegativeFloat">
  <xsd:restriction base="xsd:float">
    <xsd:minInclusive value="0"/>
  </xsd:restriction>
</xsd:simpleType>

<!--
  Positive float data type
-->
<xsd:simpleType name="positiveFloat">
  <xsd:restriction base="xsd:float">
    <xsd:minExclusive value="0"/>
  </xsd:restriction>
</xsd:simpleType>

<!--
  Positive integer data type
-->
<xsd:simpleType name="positiveInteger">
  <xsd:restriction base="xsd:integer">
    <xsd:minExclusive value="0"/>
  </xsd:restriction>
</xsd:simpleType>

<!--
  description tags
-->
<xsd:complexType name="descriptionType">
  <xsd:attribute name="field" type="xsd:string" use="required"/>
  <xsd:attribute name="value" type="xsd:string" use="required"/>
</xsd:complexType>

<!--
  yes/no data type
-->
```

```
  <xsd:simpleType name="yesNo">
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="yes"/>
      <xsd:enumeration value="no"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:schema>
```

3d.xsd

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<!--
  XML Schema definition for the "3d.xml" file defined by the Assomac
  JANE standard.

  This document conforms to the revision 0.7.0 of the standard document.
-->
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="root3d">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="geometry" maxOccurs="unbounded">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element name="description" type="descriptionType"
                                          minOccurs="0" maxOccurs="unbounded"/>
              <xsd:element name="coordinatesTransformation"
                                          type="coordinatesTransformationType"
                                          minOccurs="0" maxOccurs="unbounded"/>
              <xsd:element name="sample" type="sampleType" minOccurs="0"
                                                    maxOccurs="unbounded"/>
              <xsd:element name="work" type="workType" minOccurs="0"
                                                    maxOccurs="unbounded"/>
              <xsd:element name="heel" type="heelType" minOccurs="0"/>
            </xsd:sequence>
            <xsd:attribute name="id" type="xsd:string" use="required"/>
          </xsd:complexType>
        </xsd:element>
        <xsd:element name="info" minOccurs="0" maxOccurs="unbounded">
          <xsd:complexType>
            <xsd:simpleContent>
              <xsd:extension base="xsd:string">
                <xsd:attribute name="id" type="xsd:string"/>
              </xsd:extension>
            </xsd:simpleContent>
          </xsd:complexType>
        </xsd:element>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>

  <xsd:complexType name="heelType">
    <xsd:sequence>
      <xsd:element name="coordinatesTransformation"
                                        type="coordinatesTransformationType"/>
      <xsd:element name="insoleNailArea">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="insoleArea" maxOccurs="unbounded">
              <xsd:complexType>
                <xsd:sequence>
                  <xsd:element name="basePoint" minOccurs="3"
                                                      maxOccurs="unbounded">
                    <xsd:complexType>
                      <xsd:attribute name="x" type="xsd:float" use="required"/>
                      <xsd:attribute name="y" type="xsd:float" use="required"/>
                    </xsd:complexType>
```

```
            </xsd:element>
          </xsd:sequence>
          <xsd:attribute name="id" type="xsd:string" use="required"/>
          <xsd:attribute name="nailsAllowed" type="yesNo" use="required"/>
        </xsd:complexType>
      </xsd:element>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
<xsd:element name="heelGeometry">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="heelArea" maxOccurs="unbounded">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="basePoint" minOccurs="3"
                                                  maxOccurs="unbounded">
              <xsd:complexType>
                <xsd:attribute name="x" type="xsd:float" use="required"/>
                <xsd:attribute name="y" type="xsd:float" use="required"/>
              </xsd:complexType>
            </xsd:element>
          </xsd:sequence>
          <xsd:attribute name="id" type="xsd:string" use="required"/>
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="volume">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="cone">
              <xsd:complexType>
                <xsd:sequence>
                  <xsd:element name="vertex" maxOccurs="unbounded">
                    <xsd:complexType>
                      <xsd:attribute name="id" type="xsd:string"
                                                        use="required"/>
                      <xsd:attribute name="x" type="xsd:float"
                                                        use="required"/>
                      <xsd:attribute name="y" type="xsd:float"
                                                        use="required"/>
                    </xsd:complexType>
                  </xsd:element>
                </xsd:sequence>
                <xsd:attribute name="height" type="nonNegativeFloat"
                                                        use="required"/>
              </xsd:complexType>
            </xsd:element>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="nailProfile" minOccurs="0" maxOccurs="unbounded">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="nail" minOccurs="1" maxOccurs="unbounded">
              <xsd:complexType>
                <xsd:attribute name="x" type="xsd:float" use="required"/>
                <xsd:attribute name="y" type="xsd:float" use="required"/>
                <xsd:attribute name="z" type="xsd:float" use="required"/>
                <xsd:attribute name="theta" type="xsd:float"
                                                        use="required"/>
                <xsd:attribute name="phi" type="xsd:float"
                                                        use="required"/>
                <xsd:attribute name="length" type="xsd:float"
                                                        use="required"/>
              </xsd:complexType>
            </xsd:element>
          </xsd:sequence>
          <xsd:attribute name="id" type="xsd:string" use="required"/>
        </xsd:complexType>
```

```
          </xsd:element>
        </xsd:sequence>
        <xsd:attribute name="id" type="xsd:string" use="required"/>
      </xsd:complexType>
    </xsd:element>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="workType">
  <xsd:sequence>
    <xsd:element name="workPath" minOccurs="1" maxOccurs="unbounded">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element name="allowedMaterialsCombinations">
            <xsd:complexType>
              <xsd:sequence>
                <xsd:element name="materialsCombination" minOccurs="1"
                                                         maxOccurs="unbounded">
                  <xsd:complexType>
                    <xsd:attribute name="id" type="xsd:string"/>
                  </xsd:complexType>
                </xsd:element>
              </xsd:sequence>
            </xsd:complexType>
          </xsd:element>
          <xsd:element name="workPoint" minOccurs="1" maxOccurs="unbounded">
            <xsd:complexType>
              <xsd:attribute name="x" type="xsd:float" use="required"/>
              <xsd:attribute name="y" type="xsd:float" use="required"/>
              <xsd:attribute name="z" type="xsd:float" use="required"/>
              <xsd:attribute name="theta" type="xsd:float" use="required"/>
              <xsd:attribute name="phi" type="xsd:float" use="required"/>
              <xsd:attribute name="size" type="xsd:float" use="required"/>
            </xsd:complexType>
          </xsd:element>
        </xsd:sequence>
        <xsd:attribute name="id" type="xsd:string" use="required"/>
        <xsd:attribute name="label" use="required">
          <xsd:simpleType>
            <xsd:restriction base="xsd:string">
              <xsd:enumeration value="fore"/>
              <xsd:enumeration value="back"/>
              <xsd:enumeration value="left"/>
              <xsd:enumeration value="right"/>
              <xsd:enumeration value="side"/>
              <xsd:enumeration value="bottom"/>
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:attribute>
      </xsd:complexType>
    </xsd:element>
  </xsd:sequence>
  <xsd:attribute name="id" type="xsd:string" use="required"/>
  <xsd:attribute name="action" use="required">
    <xsd:simpleType>
      <xsd:restriction base="xsd:string">
        <xsd:enumeration value="card"/>
        <xsd:enumeration value="nail"/>
        <xsd:enumeration value="glue"/>
      </xsd:restriction>
    </xsd:simpleType>
  </xsd:attribute>
</xsd:complexType>

<xsd:complexType name="coordinatesTransformationType">
  <xsd:attribute name="reference" type="xsd:string" use="required"/>
  <xsd:attribute name="alpha" type="xsd:float" use="required"/>
  <xsd:attribute name="beta" type="xsd:float" use="required"/>
  <xsd:attribute name="gamma" type="xsd:float" use="required"/>
```

```
    <xsd:attribute name="offsetX" type="xsd:float" use="required"/>
    <xsd:attribute name="offsetY" type="xsd:float" use="required"/>
    <xsd:attribute name="offsetZ" type="xsd:float" use="required"/>
  </xsd:complexType>

  <xsd:complexType name="sampleType">
    <xsd:sequence>
      <xsd:element name="ring" minOccurs="2" maxOccurs="unbounded">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="surfacePoint" minOccurs="2"
                                                    maxOccurs="unbounded">
              <xsd:complexType>
                <xsd:attribute name="id" type="xsd:string" use="required"/>
                <xsd:attribute name="x" type="xsd:float" use="required"/>
                <xsd:attribute name="y" type="xsd:float" use="required"/>
                <xsd:attribute name="z" type="xsd:float" use="required"/>
                <xsd:attribute name="surface" use="required">
                  <xsd:simpleType>
                    <xsd:restriction base="xsd:string">
                      <xsd:enumeration value="top"/>
                      <xsd:enumeration value="upper"/>
                      <xsd:enumeration value="bottom"/>
                    </xsd:restriction>
                  </xsd:simpleType>
                </xsd:attribute>
              </xsd:complexType>
            </xsd:element>
          </xsd:sequence>
          <xsd:attribute name="id" type="xsd:string" use="required"/>
        </xsd:complexType>
      </xsd:element>
    </xsd:sequence>
    <xsd:attribute name="id" type="xsd:string" use="required"/>
    <xsd:attribute name="error" type="nonNegativeFloat" use="required"/>
  </xsd:complexType>

  <!--
    Non-negative float data type
  -->
  <xsd:simpleType name="nonNegativeFloat">
    <xsd:restriction base="xsd:float">
      <xsd:minInclusive value="0"/>
    </xsd:restriction>
  </xsd:simpleType>

  <!--
    description tags
  -->
  <xsd:complexType name="descriptionType">
    <xsd:attribute name="field" type="xsd:string" use="required"/>
    <xsd:attribute name="value" type="xsd:string" use="required"/>
  </xsd:complexType>


  <!--
    yes/no data type
  -->
  <xsd:simpleType name="yesNo">
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="yes"/>
      <xsd:enumeration value="no"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:schema>
```

materials.xsd

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<!--
  XML Schema definition for the "materials.xml" file defined by the Assomac
  JANE standard.

  This document conforms to the revision 0.7.0 of the standard document.
-->
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <!-- Main structure -->
  <xsd:element name="rootMaterials">
    <xsd:complexType>
      <!-- The document is compound by a sequence of material combinations -->
      <xsd:sequence>
        <xsd:element name="materialsCombination" minOccurs="0"
                                                    maxOccurs="unbounded">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element name="description" type="descriptionType"
                                        minOccurs="0" maxOccurs="unbounded"/>
              <!--
                A material combination is a sequence of
                materialClass -> material bindings
              -->
              <xsd:element name="materialDefinition" minOccurs="0"
                                                    maxOccurs="unbounded">
                <xsd:complexType>
                  <xsd:sequence>
                    <xsd:element name="description" type="descriptionType"
                                        minOccurs="0" maxOccurs="unbounded"/>
                    <xsd:element name="material">
                      <xsd:complexType>
                        <xsd:attribute name="id" type="xsd:string"
                                                          use="required"/>
                        <xsd:attribute name="thickness" type="positiveFloat"
                                                          use="required"/>
                      </xsd:complexType>
                    </xsd:element>
                  </xsd:sequence>
                  <xsd:attribute name="materialClassId" type="xsd:string"
                                                          use="required"/>
                </xsd:complexType>
              </xsd:element>
            </xsd:sequence>
            <xsd:attribute name="id" type="xsd:string" use="required"/>
          </xsd:complexType>
        </xsd:element>
        <xsd:element name="info" type="infoType" minOccurs="0"
                                                    maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>

  <!--
      Positive float data type
  -->
  <xsd:simpleType name="positiveFloat">
    <xsd:restriction base="xsd:float">
      <xsd:minExclusive value="0"/>
    </xsd:restriction>
  </xsd:simpleType>

  <!--
    description tags
  -->
  <xsd:complexType name="descriptionType">
```

```
    <xsd:attribute name="field" type="xsd:string" use="required"/>
    <xsd:attribute name="value" type="xsd:string" use="required"/>
  </xsd:complexType>

  <!--
    extended information tag
  -->
  <xsd:complexType name="infoType">
    <xsd:simpleContent>
      <xsd:extension base="xsd:string">
        <xsd:attribute name="id" type="xsd:string" use="required"/>
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>
</xsd:schema>
```

production.xsd

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<!--
  XML Schema definition for the "production.xml" file defined by the Assomac
  JANE standard.

  This document conforms to the revision 0.7.0 of the standard document.
-->
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <!-- Main structure -->
  <xsd:element name="rootProduction">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="order" type="orderType" minOccurs="0"
                                                   maxOccurs="unbounded"/>
        <xsd:element name="info" type="infoType" minOccurs="0"
                                                   maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>

  <xsd:complexType name="orderType">
    <!-- Orders are a sequence of materialCombination-quantity tuples -->
    <xsd:sequence>
      <xsd:element name="quantity">
        <xsd:complexType>
          <xsd:sequence maxOccurs="unbounded">
            <xsd:element name="geometry">
              <!--
                Quantities for each geometry are specified using:
                 - the number of complete shoes;
                 - the number of space components;
                 - the number of skipped components.
                One or more of these definitions are allowed.
              -->
              <xsd:complexType>
                <xsd:sequence>
                  <xsd:element name="additional" minOccurs="0"
                                                   maxOccurs="unbounded">
                    <xsd:complexType>
                      <xsd:attribute name="componentId" type="xsd:string"
                                                   use="required"/>
                      <xsd:attribute name="side" type="leftRight"
                                                   use="required"/>
                      <xsd:attribute name="count" type="xsd:positiveInteger"
                                                   use="required"/>
                    </xsd:complexType>
                  </xsd:element>
                  <xsd:element name="except" minOccurs="0"
                                                   maxOccurs="unbounded">
```

```
                    <xsd:complexType>
                      <xsd:attribute name="componentId" type="xsd:string"
                                                              use="required"/>
                      <xsd:attribute name="side" type="leftRight"
                                                              use="required"/>
                      <xsd:attribute name="count" type="xsd:positiveInteger"
                                                              use="required"/>
                    </xsd:complexType>
                  </xsd:element>
                </xsd:sequence>
                <xsd:attribute name="id" type="xsd:string" use="required"/>
                <xsd:attribute name="leftCount" type="xsd:nonNegativeInteger"
                                                  use="optional" default="0"/>
                <xsd:attribute name="rightCount" type="xsd:nonNegativeInteger"
                                                  use="optional" default="0"/>
              </xsd:complexType>
            </xsd:element>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="info" type="infoType" minOccurs="0"
                                                        maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="id" type="xsd:string" use="required"/>
    <xsd:attribute name="materialsCombinationId" type="xsd:string"
                                                              use="required"/>
  </xsd:complexType>

  <!--
    left/right data type
  -->
  <xsd:simpleType name="leftRight">
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="left"/>
      <xsd:enumeration value="right"/>
    </xsd:restriction>
  </xsd:simpleType>

  <!--
    extended information tag
  -->
  <xsd:complexType name="infoType">
    <xsd:simpleContent>
      <xsd:extension base="xsd:string">
        <xsd:attribute name="id" type="xsd:string" use="required"/>
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>
</xsd:schema>
```

Appendice A – Campi di descrizione standard

Sono stati individuati i seguenti campi di descrizione meritevoli di standardizzazione:

- `name`, che identifica il nome dell'entità che lo contiene;
- `baseModel`, usato nel file `general.xml` e identificante il nome (o codice) del modello base;
- `variant`, usato assieme a `baseModel` e identificante il nome (o codice) della variante geometrica (mocassino, allacciato, ...);
- `author`, che riporta il nome dell'autore del documento (eventualmente questo tag può essere ripetuto in caso di autori multipli).

Appendice B – Pratiche di utilizzo e Raccomandazioni

Pratiche di utilizzo

Garantire l'interoperabilità quando le applicazioni usano un formato che, come questo, consente di descrivere cose analoghe in molti modi diversi, impone l'obbligatorietà di alcune pratiche:

- modi: è obbligatorio gestire (interpretare e laddove opportuno esportare) il modo campionato o, limitatamente al testo, il modo semantico;
- continuità dei tratti: un tratto che descriva una linea continua (senza salti) non deve utilizzare il tag `<break/>`;
- campi di descrizione: è richiesto che, laddove per la descrizione che si vuole riportare sia previsto un campo standard, questo venga usato;
- dimensioni: sebbene sia possibile aggiungere tag per misure non esplicitamente previste in questo documento, non è lecito usare tag personalizzati per le misure previste;
- percorsi di tipo `perimeter`: ciascun componente deve includere uno e un solo percorso di questo tipo, al quale corrisponde un'azione di taglio, e tutti gli altri percorsi devono giacere all'interno di questo;
- campionature: deve essere garantito che la massima distanza tra la curva "ideale" e l'interpolazione che unisce i punti del campionamento con segmenti rettilinei sia non superiore a 100μm.

Raccomandazioni per il buon uso del formato

L'ampio grado di libertà contenuto in alcuni aspetti del formato impone la presenza di alcune raccomandazioni per il suo buon uso: pur non specificando nulla di obbligatorio, tali consigli assolvono al duplice compito di fare maggior luce sull'atto pratico dell'utilizzo del formato, e di cercare di ridurre al minimo gli inconvenienti dovuti alla cattiva interpretazione dello stesso.

- Divisione in percorsi: normalmente si dovrebbero usare percorsi diversi se sono presenti discontinuità nella geometria da rappresentare che separano parti concettualmente distinte del prodotto;

- divisione di un percorso in tratti: si consiglia di non abusare della divisione in tratti, riducendone il numero al minimo necessario. In particolare, un cambio di tratto è strettamente necessario solo laddove cambi l'informazione semantica;
- descrizioni semantiche: dovrebbero essere esportate da qualunque sistema CAD che sia in grado di salvarle nel suo formato dati nativo;
- tag `point`, flag `smooth/sharp`: questo flag è opzionale, ma si consiglia di specificarlo in tutti i punti che fanno parte di un percorso (quindi in tutti gli estremi di un'entità geometrica, nei punti di una campionatura, ma non nei punti di controllo di una curva di Bézier);
- segmenti di retta vettoriali: si consiglia di descriverli con un arco di cerchio piuttosto che con una curva di Bézier;
- tratti discontinui: in percorsi discontinui in cui sia possibile riconoscere una linea di supporto (ad esempio una fila di fori "all'inglese" che giacciono sulla stessa curva) si consiglia di usare un solo tratto discontinuo la cui descrizione proceda con ordine dall'estremo iniziale all'estremo finale del supporto, evitando di mescolare le descrizioni dei vari spezzoni per semplificare l'interpretazione dei dati;
- nome dell'archivio: si consiglia di assegnare un nome significativo al file d'archivio, nell'ottica di poter subito riconoscerne il contenuto (modello, eventuale variante) senza doverlo esaminare internamente.

Appendice C – Indicazioni sulle procedure di validazione

Questa appendice contiene alcune indicazioni su come potrebbe essere verificata la conformità dei sistemi software al presente documento di standard.

Versione del documento, versione dello standard e versione della certificazione

Le numerazioni del documento, dello standard e della certificazione sono in rapporto gerarchico tra di loro:

- la versione della certificazione ha una numerazione che prevede due numeri: major.minor (es: 1.3)
- lo standard prevede un terzo numero: major.minor.revision (es: 1.3.5)
- il presente documento è caratterizzato da quattro numeri: major.minor.revision.wording (es: 1.3.5.14)

Da sinistra a destra, i numeri hanno il seguente significato:

- il primo numero è il numero di versione maggiore (major). Esso è fissato a 0 per le versioni precedenti la promulgazione ufficiale e a 1 per quelle dalla promulgazione in poi. L'incremento del major è fatta qualora si introducessero modifiche che rompano definitivamente la compatibilità con le versioni precedenti (essenzialmente quando si introduce un nuovo standard);
- il secondo numero è il numero di versione minore (minor). Esso è incrementato quando vengono apportate modifiche allo standard ritenute critiche per la sua utilizzabilità e le procedure di certificazione si adattano di conseguenza. Non è garantita l'interoperabilità tra software certificati su versioni che differiscono a livello minor;
- il terzo numero è il numero di revisione (revision), ed è incrementato ad ogni modifica minore dello standard che non comporta un cambio nelle procedure di certificazione (che comunque non prevedono il terzo numero). Si può trattare quindi di aggiunte di componenti opzionali o comunque non critiche per l'interoperabilità;
- il quarto numero è relativo alla versione del testo descrittivo (wording), ed è incrementato ogniqualvolta il documento di descrizione dello standard viene modificato per essere più chiaro o esauriente ma senza alcuna modifica al formato dati.

Ogni numero può essere di lunghezza arbitraria (ad esempio, una piccola correzione alla versione 1.0.0.9 da luogo alla versione 1.0.0.10, non alla 1.0.1.0).

Aree di compatibilità

La certificazione di uno strumento come conforme allo standard non può prescindere dalle diverse funzioni che dallo strumento possono essere ottenute (non è ragionevole verificare la capacità di un tornio per forme di interpretare le direttive di taglio del pellame). Per questo è necessario definire diversi sottoinsiemi dello standard e ammettere la conformità solo rispetto a uno o più di questi.

Livelli di compatibilità

Anche limitando l'analisi ad un singolo ambito, è possibile che due applicazioni possano essere completamente compatibili l'una con l'altra anche se non implementano entrambe tutto quanto previsto nello standard. Ad esempio, essendo le descrizioni campionate obbligatorie per tutti i percorsi bidimensionali (escluso per quanto concerne i testi), non occorre né a chi scrive il file né a chi lo legge la capacità di interpretare una curva di Bézier.
Allo scopo di incentivare comunque l'implementazione completa dello standard (condizione necessaria affinché le possibilità di ottimizzazione dei processi offerte siano sfruttabili) è opportuno prevedere più livelli di conformità. Questi livelli devono essere concepiti in modo da garantire comunque l'interoperabilità, e costituirebbero uno strumento commerciale per la differenziazione dei prodotti.

Requisiti del processo di certificazione

Al fine di garantire equità nel trattamento di tutti gli attori, il processo di validazione deve essere quanto più possibile oggettivo, e non dipendente dalla personale valutazione di "correttezza" attribuita da una o più persone. Le indicazioni fornite in questa appendice non rispettano tale requisito, che dovrà essere tenuto in considerazione quando si riunirà il gruppo di lavoro che definirà il processo di certificazione. Particolare attenzione dovrà essere fatta all'astenersi dal valutare il pregio tecnico dell'oggetto in valutazione.
Le prove atte a verificare la conformità devono essere quanto più possibile compatibili ed esaustive. In particolare deve essere evitata la possibilità che possano essere ammesse diverse implementazioni tra loro incompatibili a causa di una differente interpretazione dello standard o a causa di una mancata verifica.

Risoluzione delle controversie

Nonostante possa essere messa la massima cura nella definizione del processo di validazione, è possibile che si possa giungere a una situazione nella quale due soggetti certificati incontrano problemi di interoperabilità.

Questa situazione può verificarsinei seguenti casi:

1. almeno uno dei soggetti ha male implementato lo standard, e l'errore non è stato evidenziato dai test di validazione;
2. lo standard non è completo, cioè non definisce con precisione l'elemento che provoca l'incompatibilità;
3. lo standard non è corretto, cioè è presente una contraddizione.

La procedura di risoluzione della controversia deve innanzi tutto collocare la questione in una delle 3 categorie sopra citate. Una volta classificato il problema è possibile procedere come segue:

1. se il problema appartiene alla prima categoria, la soluzione consiste nel chiedere ai soggetti in errore la rettifica del software. Inoltre, occorre valutare se l'errata interpretazione dello standard sia stata indotta da una descrizione inadeguata, nel qual caso viene riunito un gruppo di lavoro per il miglioramento del testo descrittivo e il rilascio di una nuova release del documento. La versione del documento vede incrementato il numero di wording: ad esempio, dalla 1.2.5.1 si passerebbe alla 1.2.5.2;
2. se il problema ha invece evidenziato una lacuna nel documento di standard, viene riunito un gruppo di lavoro che lo completi aggiungendo le specificazioni necessarie. Di fronte alla possibilità di scegliere una tra più alternative, la precedenza deve essere data alla soluzione tecnicamente migliore (per coerenza col resto dello standard, flessibilità, correttezza intrinseca, ...), mentre considerazioni di carattere diverso (implementazioni correnti, somiglianza con altri formati, ...) devono essere considerate di secondaria importanza. La procedura si conclude con il rilascio di una nuova revisione del documento di standard, il cui numero di versione è lo stesso del documento di partenza con l'eccezione del numero di revisione, che è incrementato, e del numero di wording, che è azzerato (ad esempio, dalla versione 1.2.5.2 si passerebbe alla versione 1.2.6.0);
3. se il documento dello standard presenta incoerenze viene riunito un gruppo di lavoro che identifichi la parte dello standard che introduce la contraddizione e la rimuova. La rimozione dell'incoerenza deve seguire i criteri già evidenziati al punto precedente, e si

conclude col rilascio di una nuova versione del documento di standard con numero di versione maggiore pari a 1, numero di versione minore pari a quello del documento di partenza incrementato di 1, e numeri di revisione e di wording pari a 0 (ad esempio, dalla versione 1.2.5.2 si passerebbe alla versione 1.3.0.0).

In ogni caso l'applicazione delle correzioni sui dispositivi già venduti ai clienti deve essere offerta senza costi supplementari a coloro che ne fanno richiesta. Inoltre l'ente certificatore deve per ciascun caso provvedere alla definizione di ulteriori test che evidenzino il problema al momento della verifica di conformità.

Dinamica del processo di validazione

Anche se non è possibile definire in maniera completamente generica la procedura di validazione, riteniamo che alcune linee guida possano essere di grande utilità nella definizione dettagliata delle procedure.

Certificazione dei lettori

La verifica dei soggetti che devono essere in grado di leggere correttamente un documento può essere affrontata nel seguente modo:

- viene approntata una suite di casi di test;
- al soggetto in verifica viene fornita la suite di test, e questo deve utilizzarla per una lavorazione;
- il prodotto viene inviato all'ente certificatore, che lo confronta con quanto atteso e dichiara o meno la conformità.

In caso di mancata conformità è consigliato che il certificatore metta a disposizione al candidato le informazioni in suo possesso che possono aiutarlo a risolvere le difformità riscontrate.

Certificazione degli scrittori

La certificazione degli scrittori passa attraverso le seguenti fasi:

- vengono specificati uno o più processi di lavorazione che devono comporre un progetto;
- il candidato presenta all'ente certificatore un file `.jane` contenente i dati di un progetto conforme a quanto prescritto e i risultati della lavorazione ottenuti da quel progetto (non necessariamente usando il file presentato – è ammesso che venga usato

un formato dati proprietario o comunque diverso da quello descritto in questo documento per la preparazione dei campioni);

- l'ente certificatore replica la realizzazione del progetto utilizzando il file fornito utilizzando solo macchine certificate e verifica la conformità di quanto ottenuto con quanto presentato;
- la dichiarazione di conformità è rilasciata se i campioni forniti e quanto realizzato dal certificatore sono sufficientemente simili.

Appendice D – Esempi 2D

Introduzione

Di seguito si trovano una serie di esempi commentati, di complessità crescente, che mostrano l'utilizzo del formato Assomac per la rappresentazione di geometrie piane.
Obiettivo degli esempi è aiutare lo sviluppatore a comprendere come possono essere usati gli strumenti a disposizione, pertanto si è scelto alle volte di non rispettare alcune delle pratiche consigliate laddove queste avrebbero condotto a produrre un documento che, sebbene più "corretto", sarebbe stato inadeguato ai fini didattici preposti.

Rettangolo

In questo semplice esempio viene definito un pezzo rettangolare, compreso tra l'origine (0,0) ed il punto (5,10). Sono utilizzati modi diversi, per mostrare al meglio le potenzialità. In particolare sono stati utilizzati tre tratti, di cui uno (l'ultimo) comprende due dei quattro lati del rettangolo (le pratiche consigliate vorrebbero che, in questo caso, venisse usato un solo tratto; in questo caso si è preferito procedere diversamente per poter meglio evidenziare alcuni aspetti).

```
<?xml version="1.0" encoding="ISO-8859-1"?>
```

Tag (opzionale ma raccomandato) di apertura di un documento XML, che specifica la versione di XML usata e la codifica dei caratteri.

```
<!--
  Document   : Sample 1
  Created on : 20-06-2007
  Author     : Universit&agrave; degli Studi di Pavia
             : Laboratorio di Microcalcolatori 2
  Description: Sample 1 - Rectangle
-->
```

Questi commenti sono opzionali e sono stati inseriti per facilitare la riconoscibilità dei documenti.

```
<root2d>
```

La radice del documento. È necessario che sia definito un tag di radice per la compatibilità con lo standard XML.

```
  <componentDependent id="1" materialClassId="1" rightCount="1" leftCount="1"
```

```
                              side="right" direction="45" directionTolerance="10">
  <description field="name" value="Leather rectangle"/>
</componentDependent>
```

Questo tag indica che si sta per definire un componente (l'unico di questo esempio) per quanto riguarda le sue caratteristiche non dipendenti dalla geometria. È stato scelto di assegnare un `id` numerico, ma qualunque stringa alfanumerica sarebbe stata valida

Sono inoltre specificati:

- la classe di materiale ad esso associata, da confrontare con quanto specificato nel file `materials.xml`;
- il numero di volte che questo componente è presente nella scarpa destra;
- il numero di volte che questo componente è presente nella scarpa sinistra;
- il fatto che la descrizione geometrica verrà data per la scarpa destra;
- il vettore direzione e la tolleranza su di esso (espressa in gradi), utile per i meccanismi di piazzamento automatico del pezzo.

Col tag di descrizione è anche specificato il nome del componente ("`Leather rectangle`"), e questo conclude le informazioni dipendenti dal componente.

```
<geometryDependent id="alpha" sizeUnit="eu" sizeValue="38" footLength="360">
```

Inizia la definizione di una nuova geometria. In questo primo esempio, all'interno della geometria si ha una sola istanza dell'unico componente dichiarato.

Come identificativo della geometria è stata scelta la stringa alfanumerica "`alpha`". Essa è del tutto scorrelata dagli identificativi utilizzati per identificare i componenti, e sarebbe potuta essere la stessa (nell'esempio "`1`"). Si è scelto però di differenziarli per non creare confusione.

Come attributi sono anche indicate alcune informazioni sulle dimensioni fisiche della calzatura. In particolare, è indicato che la scarpa è un "`38`" secondo l'unità di misura europea ("`eu`"), e che si aspetta venga calzata da un piede lungo `360`mm. Si noti che lo standard rende possibile indicare anche altre misure, che qui sono state omesse.

```
  <component id="1">
```

Vengono ora definite le caratteristiche geometriche del componente "`Leather rectangle`" di cui sono già state dichiarate le caratteristiche non geometriche, e di cui è qui specificata l'identificativo numerico.

```
    <perimeter id="1" action="cut" toolSide="0">
```

Il primo (e unico) percorso che definisce il componente "Leather rectangle", e che costituisce il suo perimetro esterno. Come già visto, è stato assegnato un identificativo numerico, del tutto indipendente dall'insieme degli identificativi dei componenti e delle geometrie.
Con degli attributi sono indicare le proprietà del perimetro. In particolare:

- il valore della proprietà `action` è `cut`, che indica che questo percorso va tagliato (è obbligatorio per il perimetro che l'azione associata sia `cut`);
- il valore della proprietà `toolSide` è `0`, che indica che lo strumento che deve eseguire il taglio deve posizionarsi al centro delle linee definite (questo è il valore di default, e non è necessario specificarlo).

```
        <point x="0" y="0" flags="sharp"/>
```

Comincia la definizione geometrica del percorso.
Viene definito un punto di inizio tratto nell'origine (0,0), e viene specificato che è un punto angoloso (primo vertice del rettangolo).

```
        <tract id="0-0" smoothness="high" fidelity="high">
```

L'attributo `smoothness` del tag `tract` indica che l'operazione (il taglio) deve essere della massima qualità (non deve presentare seghettature o altre imperfezioni, per quanto possibile).
L'attributo `fidelity` del tag `tract` indica che la geometria indicata deve essere seguita con la massima precisione (il taglio deve essere fatto esattamente dove indicato).
L'identificatore (`id`) è arbitrario ma deve essere univoco all'interno del percorso.

```
          <sampled>
```

La prima (e unica) definizione geometrica specificata per questo tratto è di tipo campionato.

```
            <point x="0" y="2" flags="smooth"/>
            <point x="0" y="4" flags="smooth"/>
            <point x="0" y="6" flags="smooth"/>
            <point x="0" y="8" flags="smooth"/>
```

Tutti i punti della descrizione campionata. Ricordiamo che il tratto in questione comincia nell'origine (0,0) ed è il lato di un rettangolo (quindi tutti i punti intermedi sono `smooth` e non punti angolosi). Il punto finale verrà specificato solo alla fine della descrizione del tratto.

```
          </sampled>
```

Termina la descrizione campionata del tratto.

```
        </tract>
```

Termina il tratto.

```
        <point x="0" y="10" flags="sharp"/>
```
Questo nuovo punto rappresenta sia il punto finale del vecchio tratto sia il punto iniziale del nuovo, ed è un punto angoloso (è in effetti il secondo vertice del rettangolo).

```
        <tract id="0-10" smoothness="high" fidelity="high">
```
Comincia un nuovo tratto, con le stesse richieste di qualità e precisione del vecchio.

```
          <sampled>
            <point x="2.5" y="10" flags="smooth"/>
          </sampled>
```
Un primo modo di descrizione geometrica: campionato, e con un solo punto intermedio.

```
          <vector>
            <bezier/>
          </vector>
```
Un secondo modo: vettoriale.

In particolare, è stata usata una curva di Bézier senza specificare nessun punto di controllo intermedio. Ricordando che i punti di inizio e di fine sono noti (si trovano prima e dopo il tratto), si può dedurre che tale distanza andrà coperta con un segmento (una curva di Bézier senza punti di controllo si riduce a un tratto di retta).

```
        </tract>
```
Termina anche questo secondo tratto.

```
        <point x="5" y="10" flags="sharp"/>
```
Il terzo vertice del rettangolo.

```
        <tract id="5-10" smoothness="high" fidelity="high">
```
Il terzo e ultimo tratto, che comprende i due rimanenti lati del rettangolo.

```
          <sampled>
            <point x="5" y="8" flags="smooth"/>
            <point x="5" y="6" flags="smooth"/>
            <point x="5" y="4" flags="smooth"/>
            <point x="5" y="2" flags="smooth"/>
            <point x="5" y="0" flags="sharp"/>
            <point x="4" y="0" flags="smooth"/>
            <point x="3" y="0" flags="smooth"/>
            <point x="2" y="0" flags="smooth"/>
            <point x="1" y="0" flags="smooth"/>
          </sampled>
```
Il primo modo scelto per descrivere l'ultimo tratto è quello campionato.

Da notare che il quinto punto descritto è il quarto angolo del rettangolo ed è quindi l'unico punto angoloso (`sharp`) mentre tutti gli altri punti sono interni al lato e sono quindi punto lisci (`smooth`).

```
          <vector>
            <arc bulge="0"/>
            <point x="5" y="0" flags="sharp"/>
            <bezier/>
          </vector>
```

Il secondo modo scelto per descrivere il tratto è quello geometrico.
Per il terzo lato è stato scelto di utilizzare un arco con proprietà di `bulge` impostata a zero (che riduce l'arco ad un segmento).
È poi specificato lo spigolo ed infine il quarto lato, per il quale si utilizza di nuovo una Bézier priva di punti di controllo.

```
        </tract>
```

Termina il tratto. Da notare che dopo questo tratto non è specificato un altro punto in quanto il percorso è chiuso ed il nuovo punto (nuovamente il primo angolo del rettangolo) coincide con primo punto specificato.

```
      </perimeter>
    </component>
  </geometryDependent>
</root2d>
```

Vengono infine chiusi tutti i tag che erano stati aperti. Segue il codice completo dell'esempio.

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<!--
  Document   : Sample 1
  Created on : 20-06-2007
  Author     : Universit&agrave; degli Studi di Pavia
             : Laboratorio di Microcalcolatori 2
  Description: Sample 1 - Rectangle
-->
<root2d>
  <componentDependent id="1" materialClassId="1" rightCount="1" leftCount="1"
                           side="right" direction="45" directionTolerance="10">
    <description field="name" value="Leather rectangle"/>
  </componentDependent>

  <geometryDependent id="alpha" sizeUnit="eu" sizeValue="38" footLength="360">
    <component id="1">
      <perimeter id="1" action="cut" toolSide="0">
        <point x="0" y="0" flags="sharp"/>
        <tract id="0-0" smoothness="high" fidelity="high">
          <sampled>
            <point x="0" y="2" flags="smooth"/>
            <point x="0" y="4" flags="smooth"/>
            <point x="0" y="6" flags="smooth"/>
            <point x="0" y="8" flags="smooth"/>
```

```
        </sampled>
      </tract>
      <point x="0" y="10" flags="sharp"/>
      <tract id="0-10" smoothness="high" fidelity="high">
        <sampled>
          <point x="2.5" y="10" flags="smooth"/>
        </sampled>
        <vector>
          <bezier/>
        </vector>
      </tract>
      <point x="5" y="10" flags="sharp"/>
      <tract id="5-10" smoothness="high" fidelity="high">
        <sampled>
          <point x="5" y="8" flags="smooth"/>
          <point x="5" y="6" flags="smooth"/>
          <point x="5" y="4" flags="smooth"/>
          <point x="5" y="2" flags="smooth"/>
          <point x="5" y="0" flags="sharp"/>
          <point x="4" y="0" flags="smooth"/>
          <point x="3" y="0" flags="smooth"/>
          <point x="2" y="0" flags="smooth"/>
          <point x="1" y="0" flags="smooth"/>
        </sampled>
        <vector>
          <arc bulge="0"/>
          <point x="5" y="0" flags="sharp"/>
          <bezier/>
        </vector>
      </tract>
    </perimeter>
  </component>
 </geometryDependent>
</root2d>
```

Alieno

Questo secondo esempio introduce un livello di complessità maggiore rispetto a quanto visto finora, comprendendo al suo interno la definizione di quattro percorsi. In particolare si vuole definire un pezzo tagliato di forma ellissoidale, che abbia sei segnati di cucitura (`mark`) al suo interno che disegnino approssimativamente un volto (due occhi ed una bocca stilizzati).

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<!--
    Document   : Sample 2
    Created on : 30-08-2006
    Author     : Università degli Studi di Pavia
               : Laboratorio di Microcalcolatori 2
    Description: Sample 2 - Alien
-->
<root2d>
```

Questa intestazione è equivalente a quella già vista nell'esempio 1.

```
  <componentDependent id="1" materialClassId="1" rightCount="1" leftCount="1"
                             side="right" direction="45" directionTolerance="10">
    <description field="name" value="Leather alien"/>
  </componentDependent>
```

L'intestazione dell'unico componente, con l'identificativo numerico ("1") che servirà per richiamarlo nella definizione della geometria e le proprietà già viste nell'esempio precedente. Non vengono specificate operazioni di scarnitura o ripiegatura.

```
  <geometryDependent id="1" sizeUnit="us-m" sizeValue="6.5" footLength="370">
```

Viene iniziata una nuova geometria che corrisponde alla taglia specificata. In questo esempio sono state usate le misure statunitensi per scarpe da uomo. Viene specificata anche la lunghezza del piede, espressa in millimetri.

```
    <component id="1">
```

L'inizio della definizione del componente.

```
      <!-- Face contour -->
      <perimeter id="1" action="cut">
```

Il primo percorso ad essere definito è il contorno del volto, di tipo `perimeter` . Come si può notare ad esso è associata l'azione del taglio (obbligatorio). È il percorso più esterno; gli altri tre percorsi saranno al suo interno.

```
        <point x="0" y="0" flags="smooth"/>
        <tract id="0-0" smoothness="high" fidelity="high">
          <sampled>
```

```
          <point x="0.28" y="3.11" flags="smooth"/>
          <!--           ...            -->
          <!-- a lot of samples omitted -->
          <!--           ...            -->
          <point x="0.28" y="-3.11" flags="smooth"/>
        </sampled>
```

Una prima definizione campionata del contorno: per compattezza sono stati omessi tutti i punti del campionamento tranne il primo e l'ultimo. Nessuno dei punti è angoloso (sono tutti `smooth`) e questo è ragionevole nella definizione di una figura ellissoidale.

```
        <vector>
          <bezier>
            <point x="0" y="7"/>
            <point x="3" y="10"/>
            <point x="7" y="10"/>
            <point x="10" y="7"/>
          </bezier>
          <point x="10.0" y="0.00" flags="smooth"/>
          <bezier>
            <point x="10" y="-7"/>
            <point x="7" y="-10"/>
            <point x="3" y="-10"/>
            <point x="0" y="-7"/>
          </bezier>
        </vector>
      </tract>
    </perimeter>
```

Viene data anche una definizione vettoriale, in cui la figura è ottenuta raccordano due linee di Bézier. Questo conclude il primo percorso, ovvero ilperimetro del volto.

```
    <!-- Mouth -->
    <closedPath id="2" action="mark">
      <point x="2" y="-3" flags="sharp"/>
      <tract id="2--3" smoothness="high" fidelity="high">
        <sampled>
          <point x="2.0" y="-3.0" flags="smooth"/>
          <!--           ...            -->
          <!-- a lot of samples omitted -->
          <!--           ...            -->
          <point x="4" y="-2.5" flags="smooth"/>
        </sampled>
        <vector>
          <arc bulge="-1"/>
          <point x="6" y="-2" flags="sharp"/>
          <arc bulge="0"/>
        </vector>
      </tract>
    </closedPath>
```

In questa parte è stato definito il percorso numero due, associato all'azione `mark` (segnato), che rappresenta la bocca. Tale percorso è chiuso (`closedPath`) ed è stato definito in maniera campionata (anche qui sono stati omessi molti dei punti campionati, per brevità) ed in maniera vettoriale utilizzando due archi:

- un arco tra il punto (2,-3) ed il punto (6,-2), con parametro `bulge` pari a "-1", che significa che l'arco è una semicirconferenza;

- un arco tra il punto (6,-2) ed il punto (2,-3), con parametro `bulge` pari a zero, che significa che l'arco si riduce ad un segmento.

Ricordiamo che nei percorsi chiusi il primo punto ad essere specificato è sia estremo iniziale per la prima specifica geometrica che estremo finale dell'ultima.

```
    <!-- Left eye -->
    <closedPath id="3" action="mark">
      <point x="2" y="5" flags="smooth"/>
      <tract id="2-5" smoothness="high" fidelity="high">
        <sampled>
          <point x="2.486" y="6.04805" flags="smooth"/>
          <!--           ...            -->
          <!-- a lot of samples omitted -->
          <!--           ...            -->
          <point x="2.486" y="4.28045" flags="smooth"/>
        </sampled>
        <vector>
          <bezier>
            <bezierControlPoint x="2" y="8"/>
            <bezierControlPoint x="8" y="6"/>
            <bezierControlPoint x="8" y="4"/>
            <bezierControlPoint x="2" y="3"/>
          </bezier>
        </vector>
      </tract>
    </closedPath>
```

Il terzo percorso permette di disegnare l'occhio sinistro. Viene definito in maniera campionata e vettoriale, utilizzando una curva di Bézier chiusa che inizia e termina nel punto (2,5).

```
    <!-- Right eye -->
    <closedPath id="4" action="mark">
      <point x="7.5" y="6" flags="smooth"/>
      <tract id="7.5-6" smoothness="high" fidelity="high">
        <sampled>
          <point x="7.500675" y="3.5457" flags="smooth"/>
          <!--           ...            -->
          <!-- a lot of samples omitted -->
          <!--           ...            -->
          <point x="7.992075" y="5.5113" flags="smooth"/>
        </sampled>
        <vector>
          <bezier>
            <bezierControlPoint x="6" y="6"/>
            <bezierControlPoint x="7.5" y="0"/>
            <bezierControlPoint x="7.5" y="0"/>
            <bezierControlPoint x="9" y="6"/>
          </bezier>
        </vector>
      </tract>
    </closedPath>
```

Il quarto ed ultimo percorso permette di disegnare l'occhio destro. Come l'altro occhio, viene definito sia in modo campionato che vettoriale, utilizzando una curva di Bézier chiusa che inizia e termina nel punto (7.5,6).

```
  </component>
</geometryDependent>
```

```
</root2d>
```

Non è necessario aggiungere altro e si chiudono tutti i tag rimasti aperti.

Segue il codice completo dell'esempio.

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<!--
    Document   : Sample 2
    Created on : 20-06-2007
    Author     : UniversitÃ  degli Studi di Pavia
               : Laboratorio di Microcalcolatori 2
    Description: Sample 2 - Alien
-->
<root2d>
  <componentDependent id="1" materialClassId="1" rightCount="1" leftCount="1"
                      side="right" direction="45" directionTolerance="10">
    <description field="name" value="Leather alien"/>
  </componentDependent>

  <geometryDependent id="1" sizeUnit="us-m" sizeValue="6.5" footLength="370">
    <component id="1">
      <!-- Face contour -->
      <perimeter id="1" action="cut">
        <point x="0" y="0" flags="smooth"/>
        <tract id="0-0" smoothness="high" fidelity="high">
          <sampled>
            <point x="0.28" y="3.11" flags="smooth"/>
            <point x="1.04" y="5.47" flags="smooth"/>
            <point x="2.16" y="7.13" flags="smooth"/>
            <point x="3.52" y="8.11" flags="smooth"/>
            <point x="5.00" y="8.44" flags="smooth"/>
            <point x="6.48" y="8.11" flags="smooth"/>
            <point x="7.84" y="7.13" flags="smooth"/>
            <point x="8.96" y="5.47" flags="smooth"/>
            <point x="9.72" y="3.11" flags="smooth"/>
            <point x="10.0" y="0.00" flags="smooth"/>
            <point x="9.72" y="-3.11" flags="smooth"/>
            <point x="8.96" y="-5.47" flags="smooth"/>
            <point x="7.84" y="-7.13" flags="smooth"/>
            <point x="6.48" y="-8.11" flags="smooth"/>
            <point x="5.00" y="-8.44" flags="smooth"/>
            <point x="3.52" y="-8.11" flags="smooth"/>
            <point x="2.16" y="-7.13" flags="smooth"/>
            <point x="1.04" y="-5.47" flags="smooth"/>
            <point x="0.28" y="-3.11" flags="smooth"/>
          </sampled>
          <vector>
            <bezier>
              <bezierControlPoint x="0" y="7"/>
              <bezierControlPoint x="3" y="10"/>
              <bezierControlPoint x="7" y="10"/>
              <bezierControlPoint x="10" y="7"/>
            </bezier>
            <point x="10.0" y="0.00" flags="smooth"/>
            <bezier>
              <bezierControlPoint x="10" y="-7"/>
              <bezierControlPoint x="7" y="-10"/>
              <bezierControlPoint x="3" y="-10"/>
              <bezierControlPoint x="0" y="-7"/>
            </bezier>
          </vector>
        </tract>
      </perimeter>

      <!-- Mouth -->
      <closedPath id="2" action="mark">
```

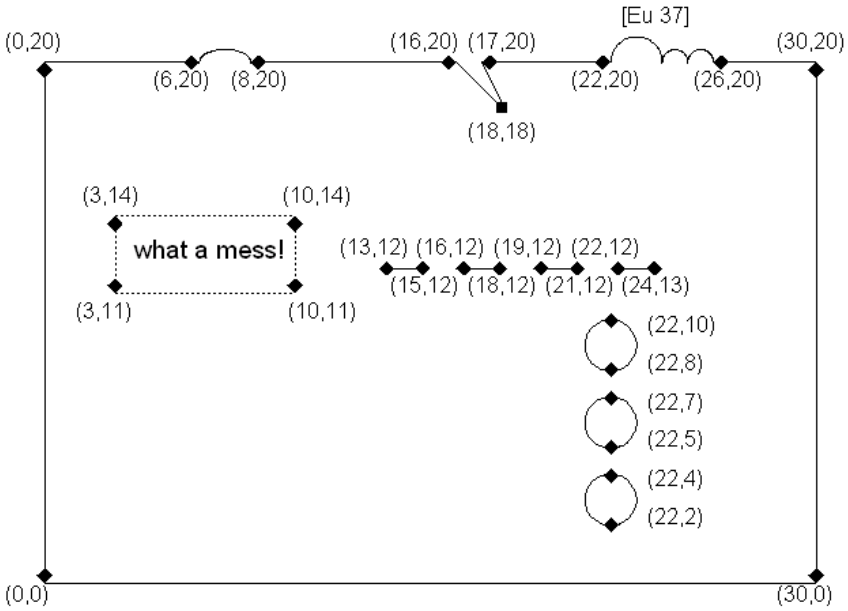
```
      <point x="2" y="-3" flags="sharp"/>
      <tract id="2--3" smoothness="high" fidelity="high">
        <sampled>
          <point x="2.0" y="-3.0" flags="smooth"/>
          <point x="2.9393399" y="-4.267767" flags="smooth"/>
          <point x="4.5" y="-4.5" flags="smooth"/>
          <point x="5.767767" y="-3.5606601" flags="smooth"/>
          <point x="6.0" y="-2.0" flags="sharp"/>
          <point x="4" y="-2.5" flags="smooth"/>
        </sampled>
        <vector>
          <arc bulge="-1"/>
          <point x="6" y="-2" flags="sharp"/>
          <arc bulge="0"/>
        </vector>
      </tract>
    </closedPath>

    <!-- Left eye -->
    <closedPath id="3" action="mark">
      <point x="2" y="5" flags="smooth"/>
      <tract id="2-5" smoothness="high" fidelity="high">
        <sampled>
          <point x="2.486" y="6.04805" flags="smooth"/>
          <point x="3.536" y="6.3696" flags="smooth"/>
          <point x="4.646" y="6.20015" flags="smooth"/>
          <point x="5.456" y="5.7392" flags="smooth"/>
          <point x="5.75" y="5.15625" flags="smooth"/>
          <point x="5.456" y="4.5968" flags="smooth"/>
          <point x="4.646" y="4.18835" flags="smooth"/>
          <point x="3.536" y="4.0464" flags="smooth"/>
          <point x="2.486" y="4.28045" flags="smooth"/>
        </sampled>
        <vector>
          <bezier>
            <bezierControlPoint x="2" y="8"/>
            <bezierControlPoint x="8" y="6"/>
            <bezierControlPoint x="8" y="4"/>
            <bezierControlPoint x="2" y="3"/>
          </bezier>
        </vector>
      </tract>
    </closedPath>

    <!-- Right eye -->
    <closedPath id="4" action="mark">
      <point x="7.5" y="6" flags="smooth"/>
      <tract id="7.5-6" smoothness="high" fidelity="high">
        <sampled>
          <point x="7.500675" y="3.5457" flags="smooth"/>
          <point x="7.5096" y="2.0064" flags="smooth"/>
          <point x="7.542525" y="1.1931" flags="smooth"/>
          <point x="7.6152" y="0.9888" flags="smooth"/>
          <point x="7.734375" y="1.3125" flags="smooth"/>
          <point x="7.8888" y="2.0832" flags="smooth"/>
          <point x="8.040225" y="3.1839" flags="smooth"/>
          <point x="8.1144" y="4.4256" flags="smooth"/>
          <point x="7.992075" y="5.5113" flags="smooth"/>
        </sampled>
        <vector>
          <bezier>
            <bezierControlPoint x="6" y="6"/>
            <bezierControlPoint x="7.5" y="0"/>
            <bezierControlPoint x="7.5" y="0"/>
            <bezierControlPoint x="9" y="6"/>
          </bezier>
        </vector>
      </tract>
    </closedPath>
```

```
    </component>
  </geometryDependent>
</root2d>
```

Mess

In questo esempio si introducono altre tipologie descrittive che finora non sono state ancora usate.

In particolare, si vuole realizzare il componente mostrato in figura:



Componente realizzato nell'esempio Mess

dove:

- la linea tratteggiata è incisa
- i cerchi sono fori
- la scritta è un segnato

Si evidenziano di seguito i punti più significativi, ritenendo che il codice prodotto (riportato in fondo) sia a questo punto comprensibile.

```
RIGA 19:
  <perimeter id="1" action="cut" toolSide="1">   <!-- *** toolSide=1 *** -->
```

Il percorso numero uno è il perimetro esterno (`perimeter`). Come tale è associato all'azione di taglio, e si è scelto di dare al tag `toolside` il valore 1: significa che l'attrezzo che esegue il taglio deve stare a sinistra del percorso che, dato il verso di percorrenza scelto (orario), implica che la punta dell'attrezzo debba stare esterna al pezzo.

```
RIGA: 20
        <!-- ######################################################## -->
        <!-- #                    FIRST SIDE                        # -->
        <!-- ######################################################## -->
```

Per il solo percorso esterno sono stati messi dei commenti ben marcati che evidenziano la

definizione dei singoli lati (con il primo estremo incluso, l'ultimo escluso).

```
RIGA: 41
       <!-- first notch -->
       <tract id="6-20" smoothness="high" fidelity="low"> <!-- low fidelity -->
         <sampled>
           <point x="6.1990657" y="20.20969" flags="smooth"/>
           <point x="6.440983" y="20.368034" flags="smooth"/>
           <point x="6.712809" y="20.46656" flags="smooth"/>
           <point x="7.0" y="20.5" flags="smooth"/>
           <point x="7.287191" y="20.46656" flags="smooth"/>
           <point x="7.559017" y="20.368034" flags="smooth"/>
           <point x="7.8009343" y="20.20969" flags="smooth"/>
         </sampled>
         <vector>
           <arc bulge="0.5"/>
         </vector>
         <referenceNotch side="left" depth="0.5" angle="0"/>
       </tract>
```

La prima tacca ha tre diverse descrizioni: il modo campionato (i cui campioni sono stati omessi per brevità); il modo vettoriale la descrive come un arco di `bulge` pari a 0.5; il modo semantico la descrive come una tacca di riferimento (`referenceNotch`) i cui confini sono ricavati dalla descrizione vettoriale. Da notare i valori di qualità e precisione del tratto:

- `smoothness = "high"` implica che la tacca deve essere tagliata con precisione perché se il taglio è troppo ruvido si potrebbe non capire più il valore rappresentato;
- `fidelity = "low"` indica che, pur stando nei limiti del tratto, la tacca può non avere precisamente la forma specificata (si lascia quindi una certa libertà per quanto riguarda le indicazioni geometriche).

Lo stesso vale per la seconda tacca, anch'essa di riferimento, ma di forma triangolare (non riportata, si veda il codice completo più avanti).

```
RIGA 85:
       <!-- third notch -->
       <tract id="22-20" smoothness="high" fidelity="low"><!-- low fidelity -->
         <sampled>
           <point x="23" y="21" flags="smooth"/>
           <point x="24" y="20" flags="sharp"/>
           <point x="24.5" y="20.5" flags="smooth"/>
           <point x="25" y="20" flags="sharp"/>
           <point x="25.5" y="20.5" flags="smooth"/>
         </sampled>
         <symbolicNotch side="left" allowOpposite="yes"/>
       </tract>
```

La terza tacca è simbolica e ne vengono date solo le descrizioni semantica e vettoriale. Vale quanto detto sulle altre tacche per quanto riguarda qualità e precisione.

```
RIGA 127:
     <!-- text box -->
     <openPath id="2" action="mark" toolSide="0">
```

Il secondo percorso definisce il testo, associato all'azione segnato.

```
RIGA 129:
        <point x="3" y="11"/>
        <tract id="3-11" smoothness="low" fidelity="high"> <!-- low smooth. -->
          <semanticText height="3" text="What a mess!"/>
        </tract>
        <point x="10" y="11"/>
```

Il testo è definito utilizzando solo il modo semantico: si noti che solo nel caso del testo semantico è consentito omettere la definizione campionata. I due estremi del tratto definiscono la base di un rettangolo e nel tag `semanticText` è precisata l'altezza. Il testo deve essere posizionato all'interno di tale rettangolo.

```
RIGA 136:
      <!-- etched dashed line -->
      <openPath id ="3" action="etch" toolSide="0">
```

Il percorso tre è una linea incisa e tratteggiata.

```
RIGA 138:
        <point x="13" y="12" flags="sharp"/>
        <tract id="13-12" smoothness="high" fidelity="high">
          <sampled>
            <point x="15" y="12" flags="sharp"/>
            <break/>
            <point x="16" y="12" flags="sharp"/>
            <point x="18" y="12" flags="sharp"/>
            <break/>
            <!--           ...            -->
            <!-- a lot of samples omitted -->
            <!--           ...            -->
          </sampled>
          <vector>
            <bezier/>
            <point x="15" y="12" flags="sharp"/>
            <break/>
            <point x="16" y="12" flags="sharp"/>
            <bezier/>
```

La linea tratteggiata è ottenuta grazie al tag `<break/>`, sia nella definizione campionata sia nella definizione vettoriale (alternando, all'interno dello stesso tratto, entità Bézier di grado zero a salti).

La definizione vettoriale non è riportata per intero.

```
RIGA 170:
      <!-- circles line -->
      <openPath id="4" action="cut" toolSide="1">
```

Il percorso quattro è una fila di fori circolari.

```
RIGA 172:
        <point x="22" y="2" flags="smooth"/>
        <tract id="22-2" smoothness="high" fidelity="high">
          <sampled>
            <!--           ...            -->
            <!-- a lot of samples omitted -->
            <!--           ...            -->
```

```
        </sampled>
        <vector>
          <circle/>
          <point x="22" y="4" flags="smooth"/>
          <break/>
          <point x="22" y="5" flags="smooth"/>
          <circle/>
          <point x="22" y="7" flags="smooth"/>
          <break/>
          <point x="22" y="8" flags="smooth"/>
          <circle/>
        </vector>
      </tract>
      <point x="22" y="10" flags="smooth"/>
```

La fila di fori è ottenuta alternando delle direttive geometriche di cerchio a dei salti.

Segue il testo completo dell'esempio

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<!--
    Document   : Sample 3
    Created on : 20-06-2007
    Author     : UniversitÃ ; degli Studi di Pavia
               : Laboratorio di Microcalcolatori 2
    Description: Sample 3 - Mess
-->
<root2d>
  <componentDependent id="1" materialClassId="1" rightCount="1" leftCount="1"
                      side="right" direction="45" directionTolerance="10">
    <description field="name" value="Leather mess"/>
  </componentDependent>

  <geometryDependent id="1" sizeUnit="eu" sizeValue="37" footLength="300">
    <component id="1">
      <!-- external contour -->
      <perimeter id="1" action="cut" toolSide="1">   <!-- *** toolSide=1 *** -->
        <!-- ######################################################## -->
        <!-- #                    FIRST SIDE                        # -->
        <!-- ######################################################## -->
        <!-- bottom left corner -->
        <point x="0" y="0" flags="sharp"/>
        <tract id="0-0" smoothness="high" fidelity="high">
          <sampled>
            <!-- top left corner -->
            <point x="0" y="20" flags="sharp"/>
          </sampled>
          <vector>
            <bezier/>
        <!-- ######################################################## -->
        <!-- #                    SECOND SIDE                       # -->
        <!-- ######################################################## -->
            <!-- top left corner -->
            <point x="0" y="20" flags="sharp"/>
            <bezier/>
          </vector>
        </tract>
        <point x="6" y="20" flags="sharp"/>
        <!-- first notch -->
        <tract id="6-20" smoothness="high" fidelity="low"> <!-- low fidelity -->
          <sampled>
            <point x="6.1990657" y="20.20969" flags="smooth"/>
            <point x="6.440983" y="20.368034" flags="smooth"/>
            <point x="6.712809" y="20.46656" flags="smooth"/>
            <point x="7.0" y="20.5" flags="smooth"/>
            <point x="7.287191" y="20.46656" flags="smooth"/>
            <point x="7.559017" y="20.368034" flags="smooth"/>
```

```
      <point x="7.8009343" y="20.20969" flags="smooth"/>
    </sampled>
    <vector>
      <arc bulge="0.5"/>
    </vector>
    <referenceNotch side="left" depth="0.5" angle="0"/>
  </tract>
  <point x="8" y="20" flags="sharp"/>
  <tract id="8-20" smoothness="high" fidelity="high">
    <sampled/>
    <vector>
      <bezier/>
    </vector>
  </tract>
  <point x="16" y="20" flags="sharp"/>
  <!-- second notch -->
  <tract id="16-20" smoothness="high" fidelity="low"><!-- low fidelity -->
    <sampled>
      <point x="18" y="18" flags="sharp"/>
    </sampled>
    <vector>
      <bezier/>
      <point x="18" y="18" flags="sharp"/>
      <bezier/>
    </vector>
    <referenceNotch side="right" depth="2" angle="37"/>
  </tract>
  <point x="17" y="20" flags="sharp"/>
  <tract id="17-20" smoothness="high" fidelity="high">
    <sampled/>
    <vector>
      <bezier/>
    </vector>
  </tract>
  <point x="22" y="20" flags="sharp"/>
  <!-- third notch -->
  <tract id="22-20" smoothness="high" fidelity="low"><!-- low fidelity -->
    <sampled>
      <point x="23" y="21" flags="smooth"/>
      <point x="24" y="20" flags="sharp"/>
      <point x="24.5" y="20.5" flags="smooth"/>
      <point x="25" y="20" flags="sharp"/>
      <point x="25.5" y="20.5" flags="smooth"/>
    </sampled>
    <symbolicNotch side="left" allowOpposite="yes"/>
  </tract>
  <point x="26" y="20" flags="sharp"/>
  <tract id="26-20" smoothness="high" fidelity="high">
    <sampled/>
    <vector>
      <bezier/>
    </vector>
  </tract>
  <!-- ######################################################## -->
  <!-- #                     THIRD SIDE                       # -->
  <!-- ######################################################## -->
  <!-- top right corner -->
  <point x="30" y="20" flags="sharp"/>
  <tract id="30-20" smoothness="high" fidelity="high">
    <sampled/>
    <vector>
      <bezier/>
    </vector>
  </tract>
  <!-- ######################################################## -->
  <!-- #                     FOURTH SIDE                      # -->
  <!-- ######################################################## -->
  <!-- low right corner -->
  <point x="30" y="0" flags="sharp"/>
```

```
    <tract id="30-0" smoothness="high" fidelity="high">
      <sampled/>
      <vector>
        <bezier/>
      </vector>
    </tract>
  </perimeter>

  <!-- text box -->
  <openPath id="2" action="mark" toolSide="0">
    <point x="3" y="11"/>
    <tract id="3-11" smoothness="low" fidelity="high"> <!-- low smooth. -->
      <semanticText height="3" text="What a mess!"/>
    </tract>
    <point x="10" y="11"/>
  </openPath>

  <!-- etched dashed line -->
  <openPath id ="3" action="etch" toolSide="0">
    <point x="13" y="12" flags="sharp"/>
    <tract id="13-12" smoothness="high" fidelity="high">
      <sampled>
        <point x="15" y="12" flags="sharp"/>
        <break/>
        <point x="16" y="12" flags="sharp"/>
        <point x="18" y="12" flags="sharp"/>
        <break/>
        <point x="19" y="12" flags="sharp"/>
        <point x="21" y="12" flags="sharp"/>
        <break/>
        <point x="22" y="12" flags="sharp"/>
      </sampled>
      <vector>
        <bezier/>
        <point x="15" y="12" flags="sharp"/>
        <break/>
        <point x="16" y="12" flags="sharp"/>
        <bezier/>
        <point x="18" y="12" flags="sharp"/>
        <break/>
        <point x="19" y="12" flags="sharp"/>
        <bezier/>
        <point x="21" y="12" flags="sharp"/>
        <break/>
        <point x="22" y="12" flags="sharp"/>
        <bezier/>
      </vector>
    </tract>
    <point x="24" y="12" flags="sharp"/>
  </openPath>

  <!-- circles line -->
  <openPath id="4" action="cut" toolSide="1">
    <point x="22" y="2" flags="smooth"/>
    <tract id="22-2" smoothness="high" fidelity="high">
      <sampled>
        <point x="23" y="3" flags="smooth"/>
        <point x="22" y="4" flags="smooth"/>
        <point x="21" y="3" flags="smooth"/>
        <point x="22" y="2" flags="smooth"/>
        <break/>
        <point x="23" y="6" flags="smooth"/>
        <point x="22" y="7" flags="smooth"/>
        <point x="21" y="6" flags="smooth"/>
        <point x="22" y="5" flags="smooth"/>
        <point x="23" y="6" flags="smooth"/>
        <break/>
        <point x="23" y="9" flags="smooth"/>
        <point x="22" y="10" flags="smooth"/>
```

```
            <point x="21" y="9" flags="smooth"/>
            <point x="22" y="8" flags="smooth"/>
            <point x="23" y="9" flags="smooth"/>
            <break/>
          </sampled>
          <vector>
            <circle/>
            <point x="22" y="4" flags="smooth"/>
            <break/>
            <point x="22" y="5" flags="smooth"/>
            <circle/>
            <point x="22" y="7" flags="smooth"/>
            <break/>
            <point x="22" y="8" flags="smooth"/>
            <circle/>
          </vector>
        </tract>
        <point x="22" y="10" flags="smooth"/>
      </openPath>
    </component>
  </geometryDependent>
</root2d>
```

Componenti e taglie

Questo esempio mostra come vengono gestiti più componenti e più taglie. Viene anche mostrato come sia possibile riutilizzare la stessa geometria per taglie diverse.

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<!--
  Document   : Sample 4
  Created on : 30-08-2006
  Author     : Università degli Studi di Pavia
             : Laboratorio di Microcalcolatori 2
  Description: Sample 4 - Resizing
-->
<root2d>
```

La solita intestazione.

```
  <componentDependent id="1" materialClassId="1" rightCount="1" leftCount="1"
                             side="right" direction="0" directionTolerance="10">
    <description field="name" value="Buckle"/>
  </componentDependent>
  <componentDependent id="2" materialClassId="2" rightCount="1" leftCount="1"
                             side="right" direction="0" directionTolerance="10">
    <description field="name" value="Foo"/>
  </componentDependent>
```

Dichiarazione dei componenti, questa volta 2. Notare che il campo `id` è univoco, anche se la scelta di usare numeri è del tutto arbitraria.

```
  <geometryDependent id="eu35" sizeUnit="eu" sizeValue="35" footLength="300">
```

Inizia la definizione della geometria delle due componenti per la taglia 35.

```
    <component id="1">
      <perimeter id="1" action="cut">
        <point x="0" y="0" flags="sharp"/>
        <tract id="0-0" smoothness="high" fidelity="high">
          <sampled>
            <point x="-0.059016995" y="0.6545085" flags="smooth"/>
            <point x="0.5" y="1.0" flags="smooth"/>
            <point x="1.059017" y="0.6545085" flags="smooth"/>
            <point x="1.0" y="0.0" flags="sharp"/>
            <point x="1.0" y="-4.9" flags="smooth"/>
            <point x="1" y="-5" flags="smooth"/>
            <point x="0.795015" y="-5.4095" flags="smooth"/>
            <point x="0.66048" y="-5.672" flags="smooth"/>
            <point x="0.568645" y="-5.8295" flags="smooth"/>
            <point x="0.49536" y="-5.912" flags="smooth"/>
            <point x="0.421875" y="-5.9375" flags="smooth"/>
            <point x="0.33664" y="-5.912" flags="smooth"/>
            <point x="0.237105" y="-5.8295" flags="smooth"/>
            <point x="0.13152" y="-5.672" flags="smooth"/>
            <point x="0.040735" y="-5.4095" flags="smooth"/>
            <point x="0" y="-5" flags="sharp"/>
            <point x="0" y="-0.1" flags="smooth"/>
          </sampled>
          <vector>
            <arc bulge="2"/>
```

```
          <point x="1" y="0" flags="sharp"/>
          <bezier/>
          <point x="1" y="-5" flags="smooth"/>
          <bezier>
            <bezierControlPoint x="1" y="-6"/>
            <bezierControlPoint x="0.5" y="-6"/>
            <bezierControlPoint x="0.5" y="-6"/>
            <bezierControlPoint x="0" y="-6"/>
          </bezier>
          <point x="0" y="-5" flags="smooth"/>
          <bezier/>
        </vector>
      </tract>
    </perimeter>
  </component>
```

Definizione campionata e vettoriale del primo componente (per semplicità composto di un solo tratto).

```
  <component id="2">
    <perimeter id="1" action="cut">
      <point x="0" y="0" flags="sharp"/>
      <tract id="0-0" smoothness="high" fidelity="high">
        <sampled>
          <point x="10" y="0" flags="sharp"/>
           <!--          ...             -->
           <!-- A lot of samples omitted -->
           <!--          ...             -->
           <point x="0" y="10" flags="sharp"/>
        </sampled>
        <vector>
          <bezier/>
          <point x="10" y="0" flags="sharp"/>
          <bezier/>
          <point x="10" y="5" flags="sharp"/>
          <bezier>
            <bezierControlPoint x="5" y="5"/>
            <bezierControlPoint x="5" y="5"/>
            <bezierControlPoint x="5" y="10"/>
            <bezierControlPoint x="5" y="10"/>
          </bezier>
          <point x="0" y="10" flags="sharp"/>
          <bezier/>
        </vector>
      </tract>
    </perimeter>
  </component>
```

Definizione del secondo componente, nuovamente per semplicità ridotto ad un solo tratto.

```
</geometryDependent>

<!-- 10% larger than 35 -->
<geometryDependent id="eu36" sizeUnit="eu" sizeValue="36" footLength="330">
```

Fine della definizione geometrica per la taglia 35 e inizio di descrizione per la taglia 36 (a titolo esemplificativo, il 10% più grande della 35).

```
  <!-- Use the same buckle as eu35 -->
  <component id="1">
    <sameAsGeometry id="eu35"/>
  </component>
```

In questo caso la fibbia è uguale a quella usata per le scarpe di taglia 35. Invece di descriverla nuovamente è possibile semplicemente far riferimento a quella.

```
    <component id="2">
      <perimeter id="1" action="cut">
        <point x="0" y="0" flags="sharp"/>
        <tract id="0-0" smoothness="high" fidelity="high">
          <sampled>
            <point x="11" y="0" flags="sharp"/>
            <!--          ...             -->
            <!-- A lot of samples omitted -->
            <!--          ...             -->
            <point x="0" y="11" flags="sharp"/>
          </sampled>
          <vector>
            <bezier/>
            <point x="11" y="0" flags="sharp"/>
            <bezier/>
            <point x="11" y="5.5" flags="sharp"/>
            <bezier>
              <bezierControlPoint x="5.5" y="5.5"/>
              <bezierControlPoint x="5.5" y="5.5"/>
              <bezierControlPoint x="5.5" y="11"/>
              <bezierControlPoint x="5.5" y="11"/>
            </bezier>
            <point x="0" y="11" flags="sharp"/>
            <bezier/>
          </vector>
        </tract>
      </perimeter>
    </component>
  </geometryDependent>
</root2d>
```

Definizione della geometria del secondo componente (questa volta diverso, e pertanto è stato necessario ridefinirlo completamente).

Segue il testo completo dell'esempio

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<!--
  Document   : Sample 4
  Created on : 20-06-2007
  Author     : Università  degli Studi di Pavia
             : Laboratorio di Microcalcolatori 2
  Description: Sample 4 - Resizing
-->
<root2d>
  <componentDependent id="1" materialClassId="1" rightCount="1" leftCount="1"
                             side="right" direction="0" directionTolerance="10">
    <description field="name" value="Buckle"/>
  </componentDependent>
  <componentDependent id="2" materialClassId="2" rightCount="1" leftCount="1"
                             side="right" direction="0" directionTolerance="10">
    <description field="name" value="Foo"/>
  </componentDependent>

  <geometryDependent id="eu35" sizeUnit="eu" sizeValue="35" footLength="300">
    <component id="1">
      <perimeter id="1" action="cut">
        <point x="0" y="0" flags="sharp"/>
        <tract id="0-0" smoothness="high" fidelity="high">
          <sampled>
```

```
        <point x="-0.059016995" y="0.6545085" flags="smooth"/>
        <point x="0.5" y="1.0" flags="smooth"/>
        <point x="1.059017" y="0.6545085" flags="smooth"/>
        <point x="1.0" y="0.0" flags="sharp"/>
        <point x="1.0" y="-4.9" flags="smooth"/>
        <point x="1" y="-5" flags="smooth"/>
        <point x="0.795015" y="-5.4095" flags="smooth"/>
        <point x="0.66048" y="-5.672" flags="smooth"/>
        <point x="0.568645" y="-5.8295" flags="smooth"/>
        <point x="0.49536" y="-5.912" flags="smooth"/>
        <point x="0.421875" y="-5.9375" flags="smooth"/>
        <point x="0.33664" y="-5.912" flags="smooth"/>
        <point x="0.237105" y="-5.8295" flags="smooth"/>
        <point x="0.13152" y="-5.672" flags="smooth"/>
        <point x="0.040735" y="-5.4095" flags="smooth"/>
        <point x="0" y="-5" flags="sharp"/>
        <point x="0" y="-0.1" flags="smooth"/>
      </sampled>
      <vector>
        <arc bulge="2"/>
        <point x="1" y="0" flags="sharp"/>
        <bezier/>
        <point x="1" y="-5" flags="smooth"/>
        <bezier>
          <bezierControlPoint x="1" y="-6"/>
          <bezierControlPoint x="0.5" y="-6"/>
          <bezierControlPoint x="0.5" y="-6"/>
          <bezierControlPoint x="0" y="-6"/>
        </bezier>
        <point x="0" y="-5" flags="smooth"/>
        <bezier/>
      </vector>
    </tract>
  </perimeter>
</component>

<component id="2">
  <perimeter id="1" action="cut">
    <point x="0" y="0" flags="sharp"/>
    <tract id="0-0" smoothness="high" fidelity="high">
      <sampled>
        <point x="10" y="0" flags="sharp"/>
        <point x="10" y="5" flags="sharp"/>
        <point x="7.9524" y="5.4073" flags="smooth"/>
        <point x="6.6368" y="6.3136" flags="smooth"/>
        <point x="5.8282" y="7.3589" flags="smooth"/>
        <point x="5.3376" y="8.3152" flags="smooth"/>
        <point x="5" y="9.0625" flags="smooth"/>
        <point x="4.6624" y="9.5648" flags="smooth"/>
        <point x="4.1718" y="9.8461" flags="smooth"/>
        <point x="3.3632" y="9.9664" flags="smooth"/>
        <point x="2.0476" y="9.9977" flags="smooth"/>
        <point x="0" y="10" flags="sharp"/>
      </sampled>
      <vector>
        <bezier/>
        <point x="10" y="0" flags="sharp"/>
        <bezier/>
        <point x="10" y="5" flags="sharp"/>
        <bezier>
          <bezierControlPoint x="5" y="5"/>
          <bezierControlPoint x="5" y="5"/>
          <bezierControlPoint x="5" y="10"/>
          <bezierControlPoint x="5" y="10"/>
        </bezier>
        <point x="0" y="10" flags="sharp"/>
        <bezier/>
      </vector>
    </tract>
```

```
      </perimeter>
    </component>
  </geometryDependent>

  <!-- 10% larger than 35 -->
  <geometryDependent id="eu36" sizeUnit="eu" sizeValue="36" footLength="330">
    <!-- Use the same buckle as eu35 -->
    <component id="1">
      <sameAsGeometry id="eu35"/>
    </component>

    <component id="2">
      <perimeter id="1" action="cut">
        <point x="0" y="0" flags="sharp"/>
        <tract id="0-0" smoothness="high" fidelity="high">
          <sampled>
            <point x="11" y="0" flags="sharp"/>
            <point x="11" y="5.5" flags="sharp"/>
            <point x="8.74764" y="5.94803" flags="smooth"/>
            <point x="7.30048" y="6.94496" flags="smooth"/>
            <point x="6.41102" y="8.09479" flags="smooth"/>
            <point x="5.87136" y="9.14672" flags="smooth"/>
            <point x="5.5" y="9.96875" flags="smooth"/>
            <point x="5.12864" y="10.52128" flags="smooth"/>
            <point x="4.58898" y="10.83071" flags="smooth"/>
            <point x="3.69952" y="10.96304" flags="smooth"/>
            <point x="2.25236" y="10.99747" flags="smooth"/>
            <point x="0" y="11" flags="sharp"/>
          </sampled>
          <vector>
            <bezier/>
            <point x="11" y="0" flags="sharp"/>
            <bezier/>
            <point x="11" y="5.5" flags="sharp"/>
            <bezier>
              <bezierControlPoint x="5.5" y="5.5"/>
              <bezierControlPoint x="5.5" y="5.5"/>
              <bezierControlPoint x="5.5" y="11"/>
              <bezierControlPoint x="5.5" y="11"/>
            </bezier>
            <point x="0" y="11" flags="sharp"/>
            <bezier/>
          </vector>
        </tract>
      </perimeter>
    </component>
  </geometryDependent>
</root2d>
```

Appendice E – Esempi 3D

Segue un esempio preliminare di file 3D, contenente la rappresentazione di un cilindro con 2 livelli di dettaglio. Il cilindro ha l'asse di simmetria coincidente con l'asse *z*, una base posata sul piano *xy* e l'altra parallela a tale piano a coordinata *z* pari a 10.

```
<root3d>
  <geometry id="cilindro">
    <description field="name" value="Descrizione di un cilindro"/>
    <!--
      La seguente trasformazione posiziona il cilindro in modo che l'asse di
      simmetria sia esattamente sull'asse x e che il baricentro sia nell'origine
    -->
    <coordinatesTransformation reference="anchorage" alpha="0" beta="90"
                                  gamma="0" offsetX="0" offsetY="-5" offsetZ="0"/>

    <!-- questo campionamento approssima le basi del cilindro con 8 punti -->
    <sample id="low precision sampling" error="0.15224">
      <!-- il primo anello serve per chiudere la superficie in alto -->
      <ring id="1">
        <!-- tutti i punti coincidono -->
        <surfacePoint id="1" x="0" y="0" z="10" surface="top"/>
        <surfacePoint id="2" x="0" y="0" z="10" surface="top"/>
        <surfacePoint id="3" x="0" y="0" z="10" surface="top"/>
        <surfacePoint id="4" x="0" y="0" z="10" surface="top"/>
        <surfacePoint id="5" x="0" y="0" z="10" surface="top"/>
        <surfacePoint id="6" x="0" y="0" z="10" surface="top"/>
        <surfacePoint id="7" x="0" y="0" z="10" surface="top"/>
        <surfacePoint id="8" x="0" y="0" z="10" surface="top"/>
      </ring>
      <!-- il secondo anello definisce la faccia superiore -->
      <ring id="2">
        <surfacePoint id="1" x="2" y="0" z="10" surface="top"/>
        <surfacePoint id="2" x="1.41" y="1.41" z="10" surface="top"/>
        <surfacePoint id="3" x="0" y="2" z="10" surface="top"/>
        <surfacePoint id="4" x="-1.41" y="1.41" z="10" surface="top"/>
        <surfacePoint id="5" x="-2" y="0" z="10" surface="top"/>
        <surfacePoint id="6" x="-1.41" y="-1.41" z="10" surface="top"/>
        <surfacePoint id="7" x="0" y="-2" z="10" surface="top"/>
        <surfacePoint id="8" x="1.41" y="-1.41" z="10" surface="top"/>
      </ring>
      <!-- il terzo anello taglia il cilindro a meta' -->
      <ring id="3">
        <surfacePoint id="1" x="2" y="0" z="5" surface="upper"/>
        <surfacePoint id="2" x="1.41" y="1.41" z="5" surface="upper"/>
        <surfacePoint id="3" x="0" y="2" z="5" surface="upper"/>
        <surfacePoint id="4" x="-1.41" y="1.41" z="5" surface="upper"/>
        <surfacePoint id="5" x="-2" y="0" z="5" surface="upper"/>
        <surfacePoint id="6" x="-1.41" y="-1.41" z="5" surface="upper"/>
        <surfacePoint id="7" x="0" y="-2" z="5" surface="upper"/>
        <surfacePoint id="8" x="1.41" y="-1.41" z="5" surface="upper"/>
      </ring>
      <!-- il quarto anello definisce la faccia inferiore -->
      <ring id="4">
        <surfacePoint id="1" x="2" y="0" z="0" surface="bottom"/>
        <surfacePoint id="2" x="1.41" y="1.41" z="0" surface="bottom"/>
        <surfacePoint id="3" x="0" y="2" z="0" surface="bottom"/>
        <surfacePoint id="4" x="-1.41" y="1.41" z="0" surface="bottom"/>
        <surfacePoint id="5" x="-2" y="0" z="0" surface="bottom"/>
        <surfacePoint id="6" x="-1.41" y="-1.41" z="0" surface="bottom"/>
        <surfacePoint id="7" x="0" y="-2" z="0" surface="bottom"/>
        <surfacePoint id="8" x="1.41" y="-1.41" z="0" surface="bottom"/>
      </ring>
      <!-- il quinto anello serve per chiudere la superficie in basso -->
```

```
    <ring id="5">
      <!-- tutti i punti coincidono -->
      <surfacePoint id="1" x="0" y="0" z="0" surface="bottom"/>
      <surfacePoint id="2" x="0" y="0" z="0" surface="bottom"/>
      <surfacePoint id="3" x="0" y="0" z="0" surface="bottom"/>
      <surfacePoint id="4" x="0" y="0" z="0" surface="bottom"/>
      <surfacePoint id="5" x="0" y="0" z="0" surface="bottom"/>
      <surfacePoint id="6" x="0" y="0" z="0" surface="bottom"/>
      <surfacePoint id="7" x="0" y="0" z="0" surface="bottom"/>
      <surfacePoint id="8" x="0" y="0" z="0" surface="bottom"/>
    </ring>
  </sample>
  <!--
    questo secondo campionamento approssima le basi del cilindro con 36 punti
  -->
  <sample id="high precision sampling" error="0.0761">
    <!-- il primo anello serve per chiudere la superficie in alto -->
    <ring id="1">
      <!-- tutti i punti coincidono -->
      <surfacePoint id="1" x="0" y="0" z="10" surface="top"/>
      <surfacePoint id="2" x="0" y="0" z="10" surface="top"/>
      <surfacePoint id="3" x="0" y="0" z="10" surface="top"/>
      <surfacePoint id="4" x="0" y="0" z="10" surface="top"/>
      <surfacePoint id="5" x="0" y="0" z="10" surface="top"/>
      <surfacePoint id="6" x="0" y="0" z="10" surface="top"/>
      <surfacePoint id="7" x="0" y="0" z="10" surface="top"/>
      <surfacePoint id="8" x="0" y="0" z="10" surface="top"/>
      <surfacePoint id="9" x="0" y="0" z="10" surface="top"/>
      <surfacePoint id="10" x="0" y="0" z="10" surface="top"/>
      <surfacePoint id="11" x="0" y="0" z="10" surface="top"/>
      <surfacePoint id="12" x="0" y="0" z="10" surface="top"/>
      <surfacePoint id="13" x="0" y="0" z="10" surface="top"/>
      <surfacePoint id="14" x="0" y="0" z="10" surface="top"/>
      <surfacePoint id="15" x="0" y="0" z="10" surface="top"/>
      <surfacePoint id="16" x="0" y="0" z="10" surface="top"/>
      <surfacePoint id="17" x="0" y="0" z="10" surface="top"/>
      <surfacePoint id="18" x="0" y="0" z="10" surface="top"/>
      <surfacePoint id="19" x="0" y="0" z="10" surface="top"/>
      <surfacePoint id="20" x="0" y="0" z="10" surface="top"/>
      <surfacePoint id="21" x="0" y="0" z="10" surface="top"/>
      <surfacePoint id="22" x="0" y="0" z="10" surface="top"/>
      <surfacePoint id="23" x="0" y="0" z="10" surface="top"/>
      <surfacePoint id="24" x="0" y="0" z="10" surface="top"/>
      <surfacePoint id="25" x="0" y="0" z="10" surface="top"/>
      <surfacePoint id="26" x="0" y="0" z="10" surface="top"/>
      <surfacePoint id="27" x="0" y="0" z="10" surface="top"/>
      <surfacePoint id="28" x="0" y="0" z="10" surface="top"/>
      <surfacePoint id="29" x="0" y="0" z="10" surface="top"/>
      <surfacePoint id="30" x="0" y="0" z="10" surface="top"/>
      <surfacePoint id="31" x="0" y="0" z="10" surface="top"/>
      <surfacePoint id="32" x="0" y="0" z="10" surface="top"/>
      <surfacePoint id="33" x="0" y="0" z="10" surface="top"/>
      <surfacePoint id="34" x="0" y="0" z="10" surface="top"/>
      <surfacePoint id="35" x="0" y="0" z="10" surface="top"/>
      <surfacePoint id="36" x="0" y="0" z="10" surface="top"/>
    </ring>
    <!-- il secondo anello definisce la faccia superiore -->
    <ring id="2">
      <surfacePoint id="1" x="2" y="0" z="10" surface="top"/>
      <surfacePoint id="2" x="1.97" y="0.35" z="10" surface="top"/>
      <surfacePoint id="3" x="1.88" y="0.68" z="10" surface="top"/>
      <surfacePoint id="4" x="1.73" y="1" z="10" surface="top"/>
      <surfacePoint id="5" x="1.53" y="1.29" z="10" surface="top"/>
      <surfacePoint id="6" x="1.29" y="1.53" z="10" surface="top"/>
      <surfacePoint id="7" x="1" y="1.73" z="10" surface="top"/>
      <surfacePoint id="8" x="0.68" y="1.88" z="10" surface="top"/>
      <surfacePoint id="9" x="0.35" y="1.97" z="10" surface="top"/>
      <surfacePoint id="10" x="0" y="2" z="10" surface="top"/>
      <surfacePoint id="11" x="-0.35" y="1.97" z="10" surface="top"/>
```

```
    <surfacePoint id="12" x="-0.68" y="1.88" z="10" surface="top"/>
    <surfacePoint id="13" x="-1" y="1.73" z="10" surface="top"/>
    <surfacePoint id="14" x="-1.29" y="1.53" z="10" surface="top"/>
    <surfacePoint id="15" x="-1.53" y="1.29" z="10" surface="top"/>
    <surfacePoint id="16" x="-1.73" y="1" z="10" surface="top"/>
    <surfacePoint id="17" x="-1.88" y="0.68" z="10" surface="top"/>
    <surfacePoint id="18" x="-1.97" y="0.35" z="10" surface="top"/>
    <surfacePoint id="19" x="-2" y="0" z="10" surface="top"/>
    <surfacePoint id="20" x="-1.97" y="-0.35" z="10" surface="top"/>
    <surfacePoint id="21" x="-1.88" y="-0.68" z="10" surface="top"/>
    <surfacePoint id="22" x="-1.73" y="-1" z="10" surface="top"/>
    <surfacePoint id="23" x="-1.53" y="-1.29" z="10" surface="top"/>
    <surfacePoint id="24" x="-1.29" y="-1.53" z="10" surface="top"/>
    <surfacePoint id="25" x="-1" y="-1.73" z="10" surface="top"/>
    <surfacePoint id="26" x="-0.68" y="-1.88" z="10" surface="top"/>
    <surfacePoint id="27" x="-0.35" y="-1.97" z="10" surface="top"/>
    <surfacePoint id="28" x="0" y="-2" z="10" surface="top"/>
    <surfacePoint id="29" x="0.35" y="-1.97" z="10" surface="top"/>
    <surfacePoint id="30" x="0.68" y="-1.88" z="10" surface="top"/>
    <surfacePoint id="31" x="1" y="-1.73" z="10" surface="top"/>
    <surfacePoint id="32" x="1.29" y="-1.53" z="10" surface="top"/>
    <surfacePoint id="33" x="1.53" y="-1.29" z="10" surface="top"/>
    <surfacePoint id="34" x="1.73" y="-1" z="10" surface="top"/>
    <surfacePoint id="35" x="1.88" y="-0.68" z="10" surface="top"/>
    <surfacePoint id="36" x="1.97" y="-0.35" z="10" surface="top"/>
  </ring>
  <!-- il terzo anello taglia il cilindro a meta' -->
  <ring id="3">
    <surfacePoint id="1" x="2" y="0" z="5" surface="upper"/>
    <surfacePoint id="2" x="1.97" y="0.35" z="5" surface="upper"/>
    <surfacePoint id="3" x="1.88" y="0.68" z="5" surface="upper"/>
    <surfacePoint id="4" x="1.73" y="1" z="5" surface="upper"/>
    <surfacePoint id="5" x="1.53" y="1.29" z="5" surface="upper"/>
    <surfacePoint id="6" x="1.29" y="1.53" z="5" surface="upper"/>
    <surfacePoint id="7" x="1" y="1.73" z="5" surface="upper"/>
    <surfacePoint id="8" x="0.68" y="1.88" z="5" surface="upper"/>
    <surfacePoint id="9" x="0.35" y="1.97" z="5" surface="upper"/>
    <surfacePoint id="10" x="0" y="2" z="5" surface="upper"/>
    <surfacePoint id="11" x="-0.35" y="1.97" z="5" surface="upper"/>
    <surfacePoint id="12" x="-0.68" y="1.88" z="5" surface="upper"/>
    <surfacePoint id="13" x="-1" y="1.73" z="5" surface="upper"/>
    <surfacePoint id="14" x="-1.29" y="1.53" z="5" surface="upper"/>
    <surfacePoint id="15" x="-1.53" y="1.29" z="5" surface="upper"/>
    <surfacePoint id="16" x="-1.73" y="1" z="5" surface="upper"/>
    <surfacePoint id="17" x="-1.88" y="0.68" z="5" surface="upper"/>
    <surfacePoint id="18" x="-1.97" y="0.35" z="5" surface="upper"/>
    <surfacePoint id="19" x="-2" y="0" z="5" surface="upper"/>
    <surfacePoint id="20" x="-1.97" y="-0.35" z="5" surface="upper"/>
    <surfacePoint id="21" x="-1.88" y="-0.68" z="5" surface="upper"/>
    <surfacePoint id="22" x="-1.73" y="-1" z="5" surface="upper"/>
    <surfacePoint id="23" x="-1.53" y="-1.29" z="5" surface="upper"/>
    <surfacePoint id="24" x="-1.29" y="-1.53" z="5" surface="upper"/>
    <surfacePoint id="25" x="-1" y="-1.73" z="5" surface="upper"/>
    <surfacePoint id="26" x="-0.68" y="-1.88" z="5" surface="upper"/>
    <surfacePoint id="27" x="-0.35" y="-1.97" z="5" surface="upper"/>
    <surfacePoint id="28" x="0" y="-2" z="5" surface="upper"/>
    <surfacePoint id="29" x="0.35" y="-1.97" z="5" surface="upper"/>
    <surfacePoint id="30" x="0.68" y="-1.88" z="5" surface="upper"/>
    <surfacePoint id="31" x="1" y="-1.73" z="5" surface="upper"/>
    <surfacePoint id="32" x="1.29" y="-1.53" z="5" surface="upper"/>
    <surfacePoint id="33" x="1.53" y="-1.29" z="5" surface="upper"/>
    <surfacePoint id="34" x="1.73" y="-1" z="5" surface="upper"/>
    <surfacePoint id="35" x="1.88" y="-0.68" z="5" surface="upper"/>
    <surfacePoint id="36" x="1.97" y="-0.35" z="5" surface="upper"/>
  </ring>
  <!-- il quarto anello definisce la faccia inferiore -->
  <ring id="4">
    <surfacePoint id="1" x="2" y="0" z="0" surface="bottom"/>
    <surfacePoint id="2" x="1.97" y="0.35" z="0" surface="bottom"/>
```

```
    <surfacePoint id="3" x="1.88" y="0.68" z="0" surface="bottom"/>
    <surfacePoint id="4" x="1.73" y="1" z="0" surface="bottom"/>
    <surfacePoint id="5" x="1.53" y="1.29" z="0" surface="bottom"/>
    <surfacePoint id="6" x="1.29" y="1.53" z="0" surface="bottom"/>
    <surfacePoint id="7" x="1" y="1.73" z="0" surface="bottom"/>
    <surfacePoint id="8" x="0.68" y="1.88" z="0" surface="bottom"/>
    <surfacePoint id="9" x="0.35" y="1.97" z="0" surface="bottom"/>
    <surfacePoint id="10" x="0" y="2" z="0" surface="bottom"/>
    <surfacePoint id="11" x="-0.35" y="1.97" z="0" surface="bottom"/>
    <surfacePoint id="12" x="-0.68" y="1.88" z="0" surface="bottom"/>
    <surfacePoint id="13" x="-1" y="1.73" z="0" surface="bottom"/>
    <surfacePoint id="14" x="-1.29" y="1.53" z="0" surface="bottom"/>
    <surfacePoint id="15" x="-1.53" y="1.29" z="0" surface="bottom"/>
    <surfacePoint id="16" x="-1.73" y="1" z="0" surface="bottom"/>
    <surfacePoint id="17" x="-1.88" y="0.68" z="0" surface="bottom"/>
    <surfacePoint id="18" x="-1.97" y="0.35" z="0" surface="bottom"/>
    <surfacePoint id="19" x="-2" y="0" z="0" surface="bottom"/>
    <surfacePoint id="20" x="-1.97" y="-0.35" z="0" surface="bottom"/>
    <surfacePoint id="21" x="-1.88" y="-0.68" z="0" surface="bottom"/>
    <surfacePoint id="22" x="-1.73" y="-1" z="0" surface="bottom"/>
    <surfacePoint id="23" x="-1.53" y="-1.29" z="0" surface="bottom"/>
    <surfacePoint id="24" x="-1.29" y="-1.53" z="0" surface="bottom"/>
    <surfacePoint id="25" x="-1" y="-1.73" z="0" surface="bottom"/>
    <surfacePoint id="26" x="-0.68" y="-1.88" z="0" surface="bottom"/>
    <surfacePoint id="27" x="-0.35" y="-1.97" z="0" surface="bottom"/>
    <surfacePoint id="28" x="0" y="-2" z="0" surface="bottom"/>
    <surfacePoint id="29" x="0.35" y="-1.97" z="0" surface="bottom"/>
    <surfacePoint id="30" x="0.68" y="-1.88" z="0" surface="bottom"/>
    <surfacePoint id="31" x="1" y="-1.73" z="0" surface="bottom"/>
    <surfacePoint id="32" x="1.29" y="-1.53" z="0" surface="bottom"/>
    <surfacePoint id="33" x="1.53" y="-1.29" z="0" surface="bottom"/>
    <surfacePoint id="34" x="1.73" y="-1" z="0" surface="bottom"/>
    <surfacePoint id="35" x="1.88" y="-0.68" z="0" surface="bottom"/>
    <surfacePoint id="36" x="1.97" y="-0.35" z="0" surface="bottom"/>
  </ring>
  <!-- il quinto anello serve per chiudere la superficie in basso -->
  <ring id="5">
    <!-- tutti i punti coincidono -->
    <surfacePoint id="1" x="0" y="0" z="0" surface="bottom"/>
    <surfacePoint id="2" x="0" y="0" z="0" surface="bottom"/>
    <surfacePoint id="3" x="0" y="0" z="0" surface="bottom"/>
    <surfacePoint id="4" x="0" y="0" z="0" surface="bottom"/>
    <surfacePoint id="5" x="0" y="0" z="0" surface="bottom"/>
    <surfacePoint id="6" x="0" y="0" z="0" surface="bottom"/>
    <surfacePoint id="7" x="0" y="0" z="0" surface="bottom"/>
    <surfacePoint id="8" x="0" y="0" z="0" surface="bottom"/>
    <surfacePoint id="9" x="0" y="0" z="0" surface="bottom"/>
    <surfacePoint id="10" x="0" y="0" z="0" surface="bottom"/>
    <surfacePoint id="11" x="0" y="0" z="0" surface="bottom"/>
    <surfacePoint id="12" x="0" y="0" z="0" surface="bottom"/>
    <surfacePoint id="13" x="0" y="0" z="0" surface="bottom"/>
    <surfacePoint id="14" x="0" y="0" z="0" surface="bottom"/>
    <surfacePoint id="15" x="0" y="0" z="0" surface="bottom"/>
    <surfacePoint id="16" x="0" y="0" z="0" surface="bottom"/>
    <surfacePoint id="17" x="0" y="0" z="0" surface="bottom"/>
    <surfacePoint id="18" x="0" y="0" z="0" surface="bottom"/>
    <surfacePoint id="19" x="0" y="0" z="0" surface="bottom"/>
    <surfacePoint id="20" x="0" y="0" z="0" surface="bottom"/>
    <surfacePoint id="21" x="0" y="0" z="0" surface="bottom"/>
    <surfacePoint id="22" x="0" y="0" z="0" surface="bottom"/>
    <surfacePoint id="23" x="0" y="0" z="0" surface="bottom"/>
    <surfacePoint id="24" x="0" y="0" z="0" surface="bottom"/>
    <surfacePoint id="25" x="0" y="0" z="0" surface="bottom"/>
    <surfacePoint id="26" x="0" y="0" z="0" surface="bottom"/>
    <surfacePoint id="27" x="0" y="0" z="0" surface="bottom"/>
    <surfacePoint id="28" x="0" y="0" z="0" surface="bottom"/>
    <surfacePoint id="29" x="0" y="0" z="0" surface="bottom"/>
    <surfacePoint id="30" x="0" y="0" z="0" surface="bottom"/>
    <surfacePoint id="31" x="0" y="0" z="0" surface="bottom"/>
```

```
        <surfacePoint id="32" x="0" y="0" z="0" surface="bottom"/>
        <surfacePoint id="33" x="0" y="0" z="0" surface="bottom"/>
        <surfacePoint id="34" x="0" y="0" z="0" surface="bottom"/>
        <surfacePoint id="35" x="0" y="0" z="0" surface="bottom"/>
        <surfacePoint id="36" x="0" y="0" z="0" surface="bottom"/>
      </ring>
    </sample>
  </geometry>
</root3d>
```