



**CONSIGLIO NAZIONALE
DELLE RICERCHE**

AA.VV.

Alberto Stiglitz[^], editor

**SPECIFICHE FUNZIONALI DI ALTO
LIVELLO DEI MODULI DELLA
STAZIONE DI LAVORO
MUSICALE INTELLIGENTE
UNITA' OPERATIVA DELL'UNIVERSITA'
DEGLI STUDI DI MILANO**

**PROGETTO FINALIZZATO
SISTEMI INFORMATICI E CALCOLO PARALLELO**

SOTTOPROGETTO 7

SISTEMI DI SUPPORTO AL LAVORO INTELLETTUALE

**PROGETTO FINALIZZATO
SISTEMI INFORMATICI E CALCOLO PARALLELO**

SOTTOPROGETTO 7
Sistemi di Supporto al Lavoro Intellettuale
Coordinatore Cesare Maioli

AA.VV.
Alberto Stiglitz[^], editor

**SPECIFICHE FUNZIONALI DI ALTO
LIVELLO DEI MODULI DELLA
STAZIONE DI LAVORO
MUSICALE INTELLIGENTE
UNITA' OPERATIVA DELL'UNIVERSITA'
DEGLI STUDI DI MILANO**

N. 7/53

3/91

Rapporto Tecnico

[^] LIM - Laboratorio di Informatica Musicale
Dipartimento di Scienze dell'Informazione
Università degli Studi di Milano.

INDICE:

P0.	Introduzione.....	4
P1.	Modulo "ScoreSynth".....	4
P1.1	Generalità.....	4
P1.2	Specifiche funzionali di alto livello.....	6
P2.	Modulo "MusSer".....	7
P2.1	Generalità.....	7
P2.2	Specifiche funzionali di alto livello.....	9
P3.	Modulo "Sequencer MIDI/DSP".....	10
P3.1	Generalità.....	10
P3.2	Specifiche funzionali di alto livello.....	11
P4.	Modulo "Microprogrammi per Sequencer MIDI/DSP e Sound Accelerator Card".....	13
P4.1	Generalità.....	13
P4.2	Specifiche funzionali di alto livello.....	14
P5.	Modulo "Functional Performer".....	15
P5.1	Generalità.....	15
P5.2	Specifiche funzionali di alto livello.....	16
P6.	Modulo "TEMPER".....	18
P6.1	Generalità.....	18
P6.2	Specifiche funzionali di alto livello.....	20
P7.	Modulo "ScoreSegmenter".....	21
P7.1	Generalità.....	21
P7.2	Specifiche funzionali di alto livello.....	23
P8.	Modulo "SampleBuster".....	25
P8.1	Generalità.....	25
P8.2	Specifiche funzionali di alto livello.....	26
P9.	Modulo "Libreria di routines di lettura/scrittura di Standard MIDI File 1.0".....	28
P9.1	Generalità.....	28
P9.2	Specifiche funzionali di alto livello.....	29
P10.	Modulo "Ambiente sperimentale per microprogrammi DSP per Sound Accelerator Card".....	30
P10.1	Generalità.....	30
P10.2	Specifiche funzionali di alto livello.....	31

P11.	Interfaccia utente ipertestuale.....	33
P11.1	Generalità.....	33
P11.2	Specifiche funzionali di alto livello.....	35
P12.	Modulo "LASy Workbench".....	36
P12.1	Generalità.....	36
P12.2	Specifiche funzionali di alto livello.....	38
P13.	Modulo "Traslitteratore".....	40
P13.1	Generalità.....	40
P13.2	Specifiche funzionali di alto livello.....	41
P14.	Riferimenti bibliografici.....	43

P0. Introduzione.

Questo documento completa l'insieme dei rapporti tecnici di carattere generale relativi alla Stazione di Lavoro Musicale Intelligente (SLMI), oggetto della Linea di Ricerca Coordinata C4 MUSIC.

Nei successivi paragrafi, uno per ciascun modulo sperimentale appartenente alla SLMI sviluppato dall'unità operativa dell'Università degli Studi di Milano, vengono riportate le descrizioni dettagliate dei singoli moduli, unitamente alle specifiche funzionali di alto livello definite mediante i criteri esposti in (Camurri et al., 1989, parte II, §P8).

- Avvertenze:**
- In fase di implementazione e/o di integrazione nella SLMI, le specifiche dei singoli moduli potranno subire variazioni non sostanziali con valenza locale.
 - I nomi attribuiti a risorse e attività e utilizzati in questo documento nei diagrammi funzionali PN PT e negli scritti in pseudocodice ADA/PDL sono puramente indicativi.

P1. Modulo "ScoreSynth".

P1.1 Generalità.

Un approccio possibile all'analisi, ma anche alla composizione di brani musicali interessa l'individuazione all'interno dei brani stessi di insiemi di note che costituiscono delle unità concettuali, definite come "oggetti musicali" (Degli Antoni & Haus, 1983), e le funzioni di trasformazione che ad esse vengono applicate. Un'analisi di questo tipo produce dei modelli che descrivono le composizioni musicali esplicitandone la struttura e l'evoluzione temporale e nello stesso tempo consentendo una compattazione della descrizione senza perdita di informazione.

Un supporto scientifico a questo approccio viene fornito dalle Reti di Petri Musicali (RPM) (Camurri, Haus & Zaccaria, 1986) (Haus & Rodriguez, 1989), una particolare interpretazione di Reti di Petri (Petri, 1976) che permette di descrivere partiture musicali in termini di oggetti e trasformazioni musicali.

Il modulo *ScoreSynth* nasce quindi come strumento di supporto per l'analisi e come strumento diretto per la sintesi di partiture musicali mediante Reti di Petri (Sametti, 1988) (Scolaro, 1990) (Haus & Sametti, 1991).

ScoreSynth è un Editor/Esecutore di *Reti di Petri Musicali (RPM)*. Alla definizione delle *RPM* si è giunti associando ai nodi di tipo posto il concetto di *Oggetto Musicale* e ai nodi di tipo transizione quello di *Algoritmo* di trasformazione degli *Oggetti Musicali* stessi. Un *Oggetto Musicale* è una sequenza di messaggi MIDI (Musical Instrument Digital Interface); un *Algoritmo* è una sequenza di *Operatori Musicali* applicabili sia ai parametri che specificano una nota: altezza, intensità, durata, timbro (canale MIDI); sia all'ordine in cui le note sono disposte all'interno di un *Oggetto Musicale*.

Ciò che viene messo a disposizione da *ScoreSynth* è un ambiente integrato per la costruzione di una *RPM*, la sua esecuzione, la generazione di una partitura e il suo ascolto.

Nella fase di Editing l'utente ha a disposizione una serie di strumenti che permettono la costruzione e la modifica di una *RPM* in modalità preminentemente grafica ed interattiva. In questa fase è anche possibile definire ed eseguire i singoli *Oggetti Musicali*. Il disegno di un modello di rete è facilitato dalla possibilità di definire reti gerarchiche e reti macro.

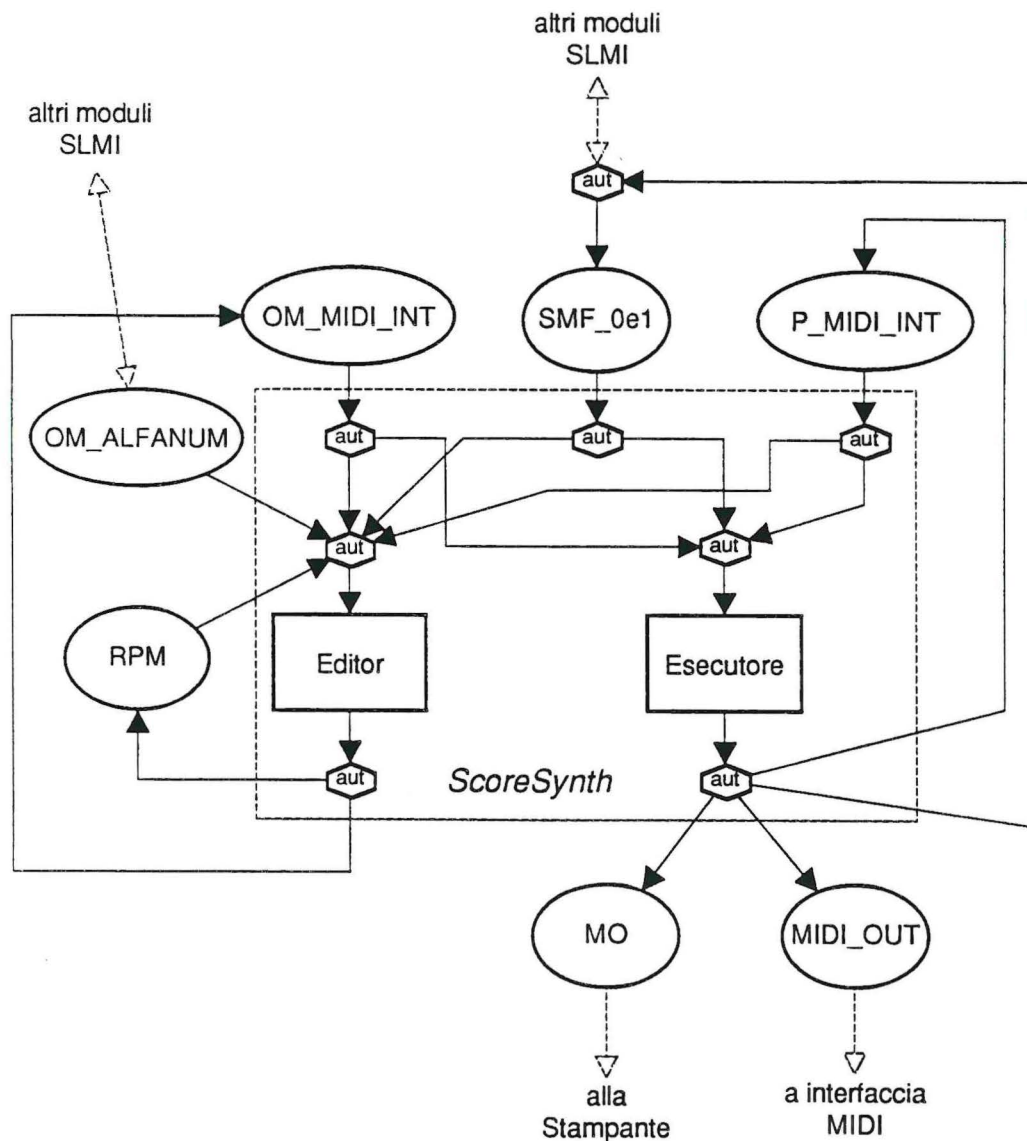
In una seconda fase, l'esecuzione di una *RPM* porta alla sintesi di una partitura MIDI e al suo eventuale ascolto differito. Durante la fase di sintesi della partitura è possibile controllare la crescita della stessa tramite un dialogo che ne rappresenta graficamente lo stato in un piano di riferimento tempo/canali_MIDI ed ispezionare tutti i valori degli attributi di posti e transizioni.

Per quanto riguarda la comunicazione con l' "esterno" si è resa possibile l'importazione e l'esportazione di file in formato Standard MIDI File (SMF) 0/1, ormai universalmente diffusi, così da garantire un'interfaccia verso tutti i programmi commerciali di sequencing e di stampa di partiture musicali. E' possibile cioè associare ad un nodo di tipo posto un *Oggetto Musicale* tramite un file in formato SMF e salvare una partitura MIDI su file in formato SMF.

Nell'ambito della SLMI, il modulo *ScoreSynth* si colloca al Livello S/W 2 (Simbolico/Strutturale) (Camurri et al., 1989, parte I, §P1) e per quanto riguarda gli ambiti operativi specifici della SLMI, *ScoreSynth* potrà essere utilizzato come strumento di ausilio alle attività di *Composizione* e *Analisi/sintesi di testi musicali* (Camurri et al., 1989, parte I, §P2).

P1.2 Specifiche funzionali di alto livello.

Diagramma funzionale.



Descrizione funzionale in pseudocodice ADA-like.

```
-- Descrizione modulo ScoreSynth

package ScoreSynth is
    procedure Editor;
    procedure Esecutore;
end Scoresynth;
```

```

-- Tipi di strutture dati utilizzati dal modulo ScoreSynth

type SMF_0e1;
--> Standard MIDI File formato 0 e 1

type OM_MIDI_INT;
--> Oggetti Musicali in un formato MIDI interno

type P_MIDI_INT;
--> Partiture MIDI in un formato interno

type RPM;
--> Reti di Petri Musicali

type OM_ALFANUM;
--> Oggetti Musicali in codifica alfanumerica

type MO;
--> Mappe ad oggetti delle partiture

type MIDI_OUT;
--> Dati standard MIDI diretti a interfaccia MIDI

```

Per quanto riguarda i dettagli implementativi e le modalità d'uso del modulo *ScoreSynth* si faccia riferimento al corrispondente documento operativo PFI-2 (M/03, 1991).

P2. Modulo "MusSer".

P2.1 Generalità.

La *musica seriale* ha rappresentato e rappresenta tuttora, per motivi storici e culturali, una delle basi concettuali più importanti per lo sviluppo della musica contemporanea. Le composizioni *seriali* si basano sull'utilizzazione di particolari oggetti musicali, composti ciascuno da dodici elementi corrispondenti a singole note della scala temperata, denominati *serie equilibrate dodecafoniche*.

Partendo da un interessante lavoro teorico (Riotte, 1963) riguardante la generazione dell'insieme delle serie equilibrate e i relativi problemi, si è definito un sottoambiente per la ricerca di pattern seriali all'interno di una composizione musicale e per la costruzione di nuovi pattern e la loro successiva elaborazione (Lazzini, 1989). Il modulo della SLMi che permette di accedere a tale sottoambiente è *MusSer*.

MusSer è un Generatore/Elaboratore/Riconoscitore/Esecutore di pattern seriali equilibrati definiti nell'ambito di scale temperate di cardinalità pari.

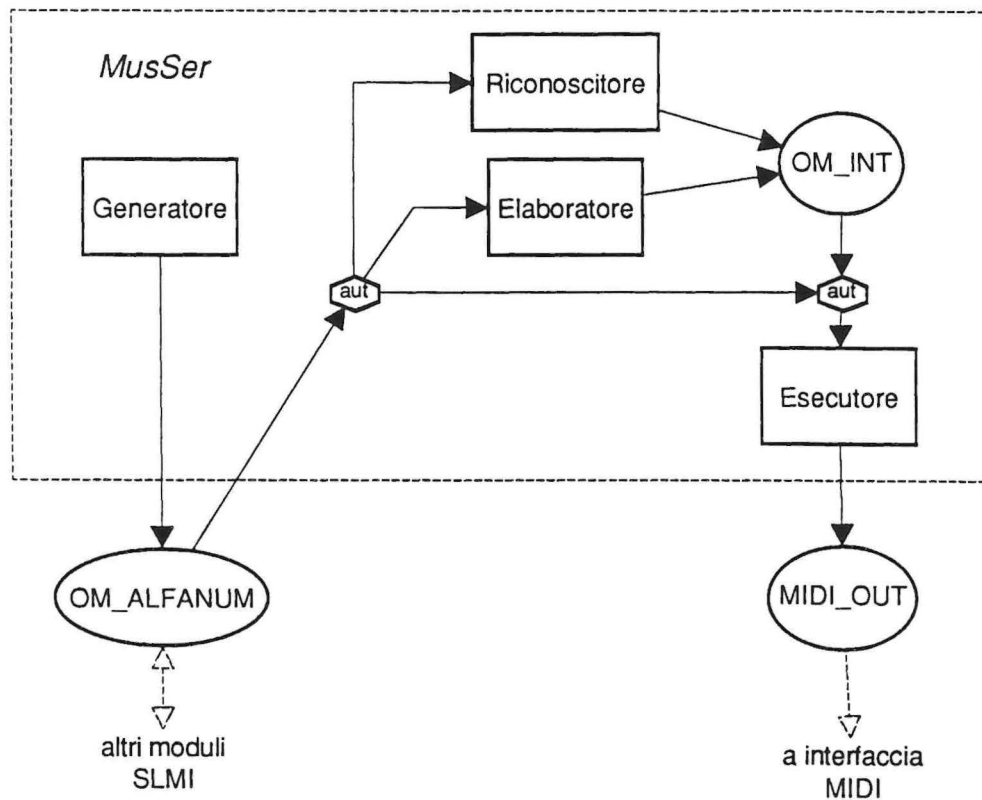
MusSer legge e scrive file in formato TabConverter (i dati musicali sono rappresentati in forma di sequenze alfanumeriche contenenti tutte le informazioni presenti usualmente nelle partiture). Questi file possono rappresentare archivi di serie dodecafoniche generati e/o elaborati da *MusSer*, oppure partiture musicali generate con altri moduli della SLMI.

Per quanto riguarda le operazioni di ricerca, queste vengono effettuate confrontando ogni serie facente parte di un archivio dodecafonico, con ciascuna delle dodici scale maggiori del sistema temperato, per verificare se sono soddisfatte alcune proprietà. In particolare sono possibili tre tipi di ricerca: il primo consente di estrarre da un file in formato TabConverter le prossime *n* serie che possono essere spezzate in sottosequenze, ciascuna delle quali contenga note che facciano tutte parte di una delle scale maggiori; il secondo permette di estrarre da un file in formato TabConverter le prossime *n* serie dodecafoniche che contengono un'unica sottoserie tonale di lunghezza specificata dall'utente, indipendentemente dal punto in cui tale sottoserie si trovi nella serie; il terzo tipo di ricerca, in modo simile al primo, consente di estrarre da un file in formato TabConverter le prossime *n* serie che possono essere spezzate in sottoserie, ciascuna delle quali contenga note che facciano tutte parte di una delle scale maggiori, dove però il modo in cui sarà spezzata la serie dodecafonica è stato specificato preventivamente dall'utente.

Nell'ambito della SLMI, il modulo *MusSer* si colloca al Livello S/W 2 (Simbolico/Strutturale) (Camurri et al., 1989, parte I, §P1) e per quanto riguarda gli ambiti operativi specifici della SLMI, *MusSer* potrà essere utilizzato come strumento di ausilio all'attività di *Composizione* (Camurri et al., 1989, parte I, §P2).

P2.2 Specifiche funzionali di alto livello.

Diagramma funzionale.



Descrizione funzionale in pseudocodice ADA-like.

-- Descrizione modulo *MusSer*

package *MusSer* **is**

procedure *Generatore*;

procedure *Riconoscitore*;

procedure *Elaboratore*;

procedure *Esecutore*;

end *MusSer*;

-- Tipi di strutture dati utilizzati dal modulo *MusSer*

type *OM_Alfanum*;

--> Oggetti Musicali in codifica alfanumerica

```
type OM_INT;  
--> Oggetti Musicali in un formato interno  
type MIDI_OUT;  
--> Dati standard MIDI diretti a interfaccia MIDI
```

Per quanto riguarda i dettagli implementativi e le modalità d'uso del modulo *MusSer* si faccia riferimento al corrispondente documento operativo PFI-2 (M/08, 1991).

P3. Modulo "Sequencer MIDI/DSP".

P3.1 Generalità.

Una delle classi di programmi più diffuse in ambito musicale interessa la generazione, la registrazione, l'elaborazione e la riproduzione di sequenze di dati MIDI. I programmi appartenenti a questa classe sono denominati *Sequencer*.

In generale i programmi sequencer sono predisposti per controllare dispositivi esterni (per la sintesi e l'elaborazione del suono) collegati all'elaboratore via interfaccia MIDI. Nell'ambito della SLMI, avendo a disposizione una scheda DSP interna all'elaboratore (Camurri et al., 1989, parte I, §P6), si è pensato di realizzare il modulo *Sequencer MIDI/DSP* in modo che fosse in grado di gestire, oltre ai dispositivi MIDI esterni, anche la scheda DSP. Tale scheda, utilizzata in altri sottoambienti della SLMI per le attività di analisi/sintesi del suono e per la costruzione di modelli timbrici (ved. §P10 e §P12), viene quindi vista da programma come un sintetizzatore a cui è possibile inviare sequenze di dati MIDI come a un qualsiasi dispositivo MIDI esterno (Ballista, 1990).

Il *Sequencer MIDI/DSP* consente di effettuare tre attività principali: registrazione, editing e riproduzione di sequenze MIDI distribuite su molteplici tracce (num max = 32). Le sequenze possono essere lette o memorizzate in forma di Standard MIDI File v.1.0. Le tracce possono essere singolarmente assegnate per pilotare l'esecuzione da parte di dispositivi MIDI esterni e/o la scheda DSP interna. E' possibile inoltre effettuare delle operazioni di mixing sulle tracce, ovvero di bilanciamento delle intensità delle note MIDI appartenenti alle tracce (Bettitoni, 1990), come generalmente avviene nel processo di registrazione multitraccia analogica. Nella SLMI questo processo invece può essere controllato

direttamente via software, operando inoltre direttamente su dati musicali numerici (quindi senza perdita di qualità o introduzione di rumori parassiti).

Dall'interno del Sequencer MIDI/DSP è anche possibile richiamare il modulo *Ambiente sperimentale per microprogrammi DSP per Sound Accelerator Card* (ved. §P10) e assegnare differenti modelli timbrici agli strumenti virtuali associati alle tracce che sono state assegnate per il controllo della scheda DSP.

Nell'ambito della SLMI, il modulo *Sequencer MIDI/DSP* si colloca in modo intermedio tra il Livello S/W 2 (Simbolico/Strutturale) e il Livello S/W 1 (Esecutivo/Operativo) (Camurri et al., 1989, parte I, §P1) e per quanto riguarda gli ambiti operativi specifici della SLMI, *Sequencer MIDI/DSP* potrà essere utilizzato come strumento di ausilio alle attività di *Orchestrazione* e di *Produzione master audio numerici* (Camurri et al., 1989, parte I, §P2).

P3.2 Specifiche funzionali di alto livello.

Diagramma funzionale.

Ved. pag. 12.

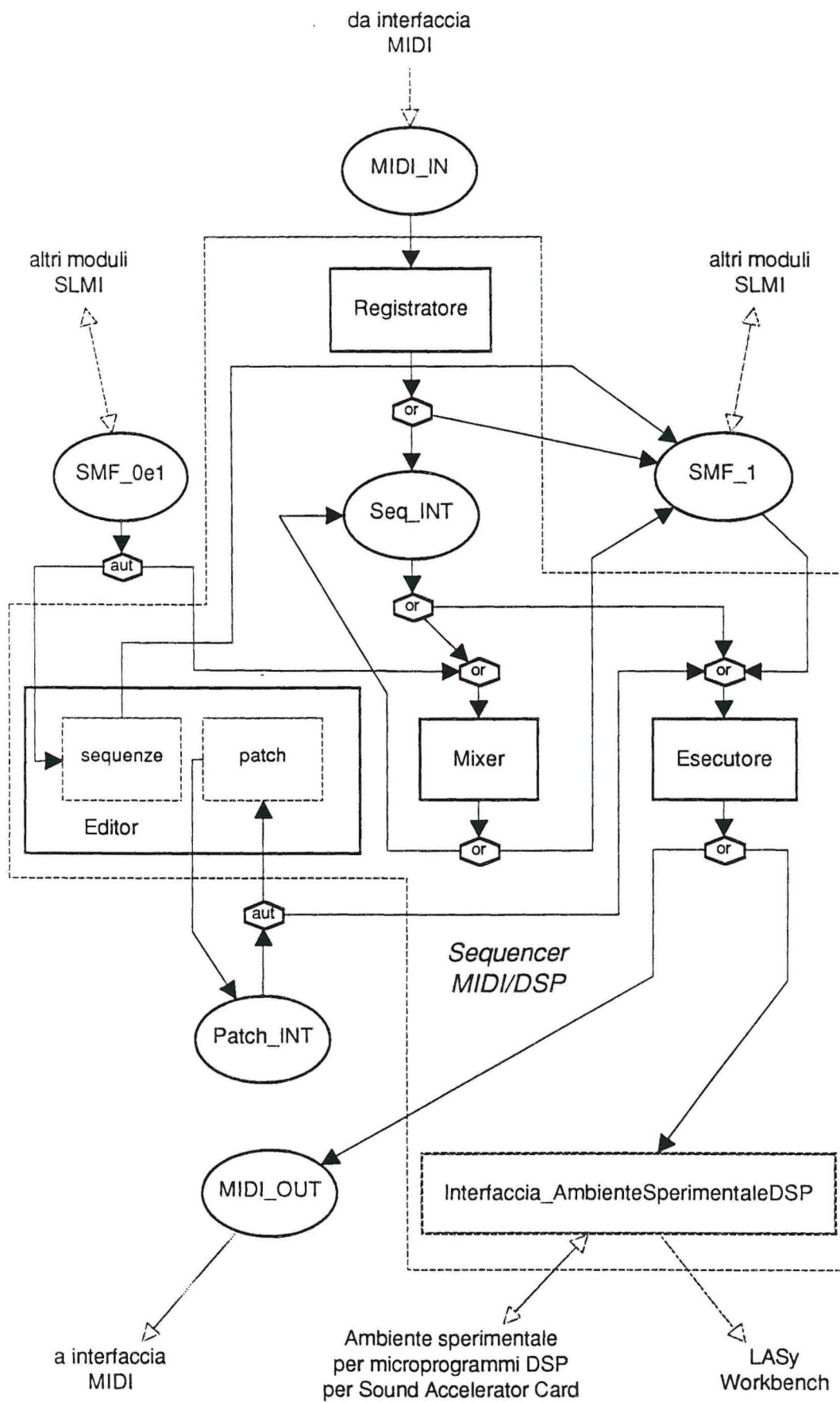
Descrizione funzionale in pseudocodice ADA-like.

```
-- Descrizione modulo Sequencer MIDI/DSP

package Sequencer_MIDI_DSP is
  procedure Registratore;
  procedure Mixer;
  procedure Editor;
  procedure Esecutore;
  procedure Interfaccia_AmbienteSperimentaleDSP;
end Sequencer_MIDI_DSP;

-- Tipi di strutture dati utilizzati dal modulo
-- Sequencer MIDI/DSP

type SMF_0e1;
--> Standard MIDI File formato 0 e 1
```



```

type SMF_1;
--> Standard MIDI File formato 1
type Seq_INT;
--> Dati sequenze in formato interno
type Patch_INT;
--> Dati patch (corrispondenze timbri <-> strumenti virtuali) in
--> formato interno
type MIDI_IN;
--> Dati standard MIDI provenienti da interfaccia MIDI
type MIDI_OUT;
--> Dati standard MIDI diretti a interfaccia MIDI

```

Per quanto riguarda i dettagli implementativi e le modalità d'uso del modulo *Sequencer MIDI/DSP* si faccia riferimento al corrispondente documento operativo PFI-2 (M/05, 1991).

P4. Modulo "Microprogrammi per Sequencer MIDI/DSP e Sound Accelerator Card".

P4.1 Generalità.

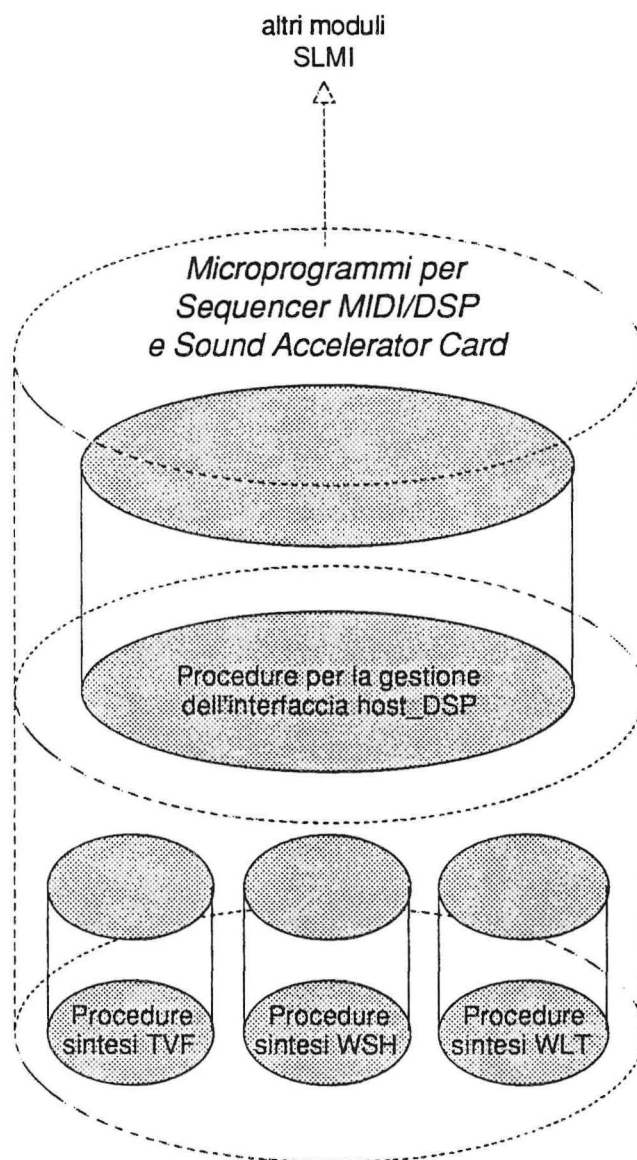
Il modulo *Microprogrammi per Sequencer MIDI/DSP e Sound Accelerator Card* è stato progettato come strumento di supporto per altri moduli della SLMI: in particolare per il modulo *Sequencer MIDI/DSP* (ved. §P3) e per il modulo *Ambiente sperimentale per microprogrammi DSP per Sound Accelerator Card* (ved. §P10). Il modulo *Microprogrammi per Sequencer MIDI/DSP e Sound Accelerator Card* consiste di due parti fondamentali: la prima comprende routine di gestione della Host Interface scritte in MPW Assembly; la seconda è costituita dall'implementazione in Assembly 56001 di tre algoritmi per la sintesi del suono mediante scheda DSP Motorola 5600. I tre algoritmi sono relativi a tre differenti tecniche di sintesi: sintesi mediante Two Variable Function (TVF) (Mitsubishi, 1982) (Borgonovo & Haus, 1986), sintesi mediante Waveshaping (WSH) (Roads, 1979) e sintesi mediante trasformata Wavelet (WLT) (Dutilleux, Grossmann & Kronland-Martinet, 1988).

Nell'ambito della SLMI, il modulo *Microprogrammi per Sequencer MIDI/DSP e Sound Accelerator Card* si colloca in modo intermedio al Livello

S/W 1 (Esecutivo/Operativo) (Camurri et al., 1989, parte I, §P1) e per quanto riguarda gli ambiti operativi specifici della SLMI, l'utilizzazione dei *Microprogrammi per Sequencer MIDI/DSP e Sound Accelerator Card* è subordinata all'uso dei moduli *Sequencer MIDI/DSP* (ved. §P3) e *Ambiente sperimentale per microprogrammi DSP per Sound Accelerator Card* (ved. §P10) e di conseguenza interessa le relative aree operative.

P4.2 Specifiche funzionali di alto livello.

Diagramma funzionale.



Descrizione funzionale in pseudocodice ADA-like.

```
-- Descrizione modulo Microprogrammi per Sequencer MIDI/DSP  
-- e Sound Accelerator Card
```

```
package Microprogrammi_DSP is  
--> procedure per la gestione dell'interfaccia host_DSP;  
--> procedure per la sintesi con Two Variable Function (TVF);  
--> procedure per la sintesi con Waveshaping (WSH);  
--> procedure per la sintesi con trasformata Wavelet (WLT);  
end Microprogrammi_DSP;
```

Per quanto riguarda i dettagli implementativi e le modalità d'uso del modulo *Microprogrammi per Sequencer MIDI-DSP e Sound Accelerator Card* si faccia riferimento al corrispondente documento operativo PFI-2 (M/10, 1991).

P5. Modulo "Functional Performer".

P5.1 Generalità.

Uno degli approcci possibili alla composizione musicale è quello di partire non più dai concetti di *singola nota* e *singolo accordo* per costruire linee melodiche e tessiture armoniche, bensì di utilizzare come unità elementari *oggetti musicali*, specificando di volta in volta le trasformazioni da eseguire su di essi. Inoltre, uno degli argomenti più attuali e interessanti, oltre che passibile di futuri sviluppi, nell'ambito dell'informatica musicale, è quello della performance in tempo reale di brani musicali eseguiti con strumenti informatici. Nell'ambito della SLMi è stato progettato il modulo *Functional Performer* che consente di adottare l'approccio compositivo a cui si è accennato sopra per comporre ed eseguire in tempo reale brani musicali.

Il modulo *Functional Performer* si configura come strumento per la performance funzionale in tempo reale di oggetti musicali (Haus & Stiglitz, 1990), assumendo come concetti fondamentali quelli di *Oggetto Musicale* e di *Operatore Musicale di Trasformazione*.

Functional Performer accetta in input una serie di frammenti musicali codificati in forma di Standard MIDI File e, sfruttando le routines ed il ciclo di lettura di SMF del modulo *Libreria di routines di lettura/scrittura di Standard MIDI File 1.0* (ved. §P9), li trasforma in un formato di rappresentazione interna.

A ciascun brano da eseguire corrisponde un archivio musicale, costituito da un insieme di frammenti musicali che deve essere caricato in memoria prima della performance. Durante l'esecuzione, sullo schermo viene visualizzata una mappa dei frammenti musicali disponibili, da cui è possibile selezionare quelli da elaborare.

Gli operatori a disposizione attuano sui frammenti melodici le seguenti trasformazioni: inversione speculare di altezze o di durate o di dinamiche, dilatazione o contrazione di intervalli o di durate o di dinamiche, trasposizione in altezza verso il registro acuto o verso il registro grave, accelerazione o rallentamento dell'esecuzione, crescendo o diminuendo di dinamiche, inversione di altezze o di durate o di dinamiche o totale, retrogradazione di altezze o di durate o di dinamiche o totale, copia di sottoframmenti, filtraggio o eliminazione di note di sottoframmenti, fusione di frammenti con allineamento a sinistra o con allineamento a destra o con centratura o con giustificazione.

I dati relativi a un'esecuzione possono essere codificati in SMF, se l'utente attiva l'appropriata funzione prima dell'inizio della performance; in questo modo i dati relativi alla partitura generata non vengono solamente inviati all'interfaccia MIDI, ma vengono anche memorizzati su disco in forma di SMF.

Nell'ambito della SLMI, il modulo *Functional Performer* si colloca al Livello S/W 2 (Simbolico/Strutturale) (Camurri et al., 1989, parte I, §P1) e per quanto riguarda gli ambiti operativi specifici della SLMI, *Functional Performer* potrà essere utilizzato come strumento di ausilio alle attività di *Performance musicale* e *Analisi/sintesi di testi musicali* (Camurri et al., 1989, parte I, §P2).

P5.2 Specifiche funzionali di alto livello.

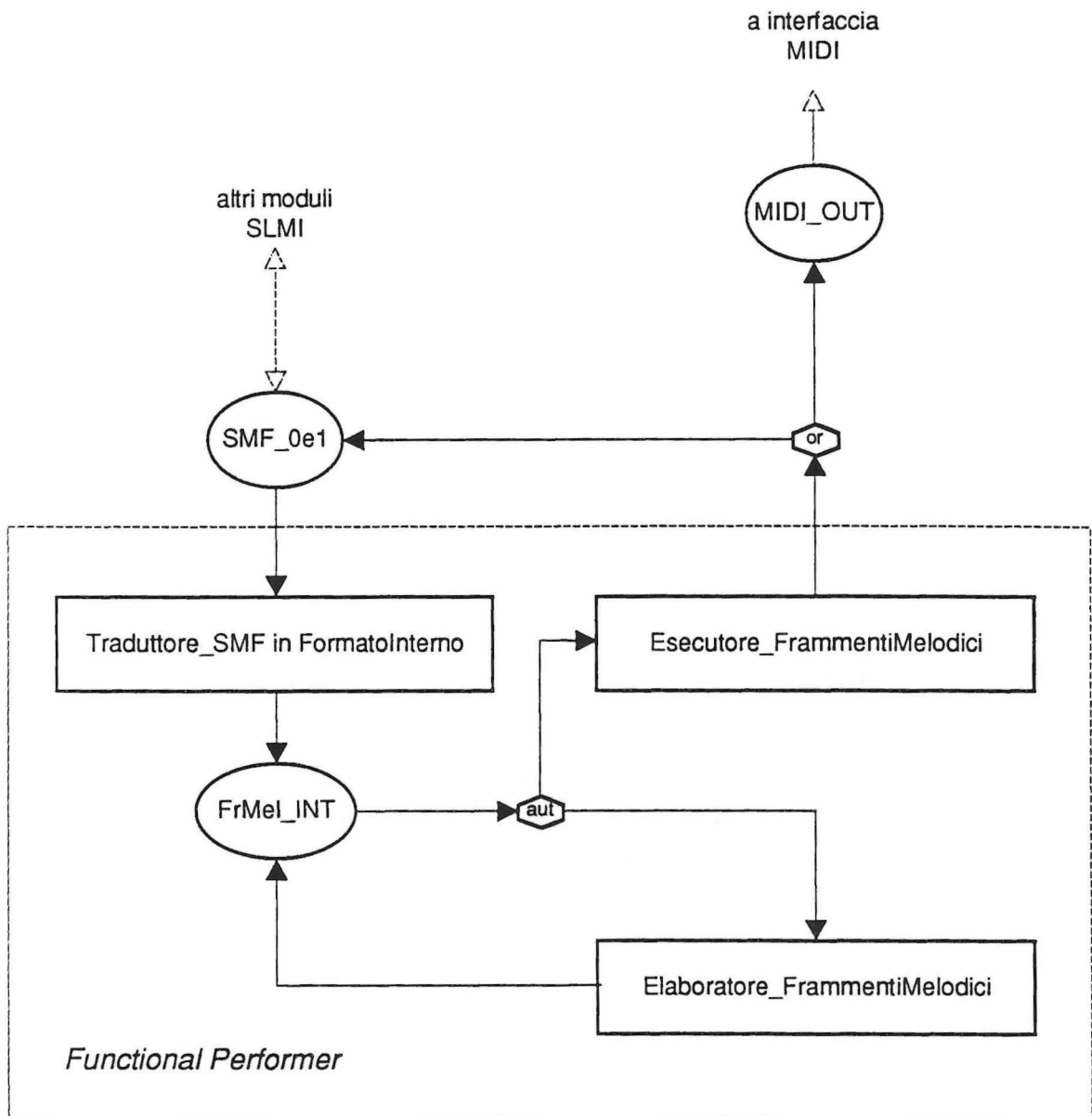
Diagramma funzionale.

Ved. pag. 17.

Descrizione funzionale in pseudocodice ADA-like.

-- Descrizione modulo *Functional Performer*

```
package Functional_Performer is  
  procedure Traduttore_SMFinFormatoInterno;  
  procedure Elaboratore_FrammentiMelodici;
```



```

    procedure Esecutore_FrammentiMelodici;
end Functional_Performer;

-- Tipi di strutture dati utilizzati dal modulo
-- Functional Performer

type SMF_0e1;
--> Standard MIDI File formato 0 e 1
type FrMel_INT;
--> Frammenti Melodici codificati in un formato interno
type MIDI_OUT;
--> Dati standard MIDI diretti a interfaccia MIDI

```

Per quanto riguarda i dettagli implementativi e le modalità d'uso del modulo *Functional Performer* si faccia riferimento al corrispondente documento operativo PFI-2 (M/23, 1991).

P6. Modulo "TEMPER".

P6.1 Generalità.

Il modulo *TEMPER* è stato studiato per tradurre in musica immagini grafiche di un tipo particolare, denominato tassellazioni (Morini, 1987) (Haus & Morini, 1991); il modulo effettua esecuzioni grafico-musicali in tempo reale.

TEMPER è suddiviso in due sezioni logiche: un *Editor grafico* e un *Esecutore*.

L'*Editor grafico* permette di costruire, registrare, riprendere e modificare tassellazioni, composte da vettori di traslazione e lati. Le tassellazioni sono organizzate in sequenze ordinate, ed ogni sequenza può essere memorizzata su disco. Saranno utilizzabili schemi di tassellazione comprendenti solo operazioni di traslazione.

L'*Esecutore* di sequenze di tassellazioni visualizza, seguendo la temporizzazione indicata, tutti i passaggi della trasformazione dalla prima tassellazione alla seconda, quindi dalla seconda alla terza e così di seguito fino alla fine della sequenza. Quando la scansione di ogni passaggio incontra uno dei punti che definiscono la linea spezzata dei lati, viene generata una nota musicale corrispondente ad una mappatura dei registri musicali nel piano di definizione delle tassellazioni.

Per permettere una maggior versatilità nella traduzione da grafica a musica sono stati resi disponibili alcuni parametri che modificano il comportamento della funzione di traduzione. I parametri, associati alle singole tassellazioni della sequenza, possono essere suddivisi in tre gruppi:

- *Parametri di tempo*, che danno la possibilità di regolare la velocità generale dell'esecuzione grafica, a cui è legata quella musicale (si può impostare il numero di passi in cui deve essere eseguita ogni trasformazione ed il tempo in cui deve essere effettuata la scansione di ogni passaggio).

- *Parametri di scala*, che permettono di decidere quale scala musicale adottare, in quale tonalità lavorare e a che intervallo musicale deve corrispondere una determinata variazione in altezza sul disegno. Solo le note appartenenti alla scala adottata sono utilizzate, consentendo di strutturare armonicamente il brano musicale generato.

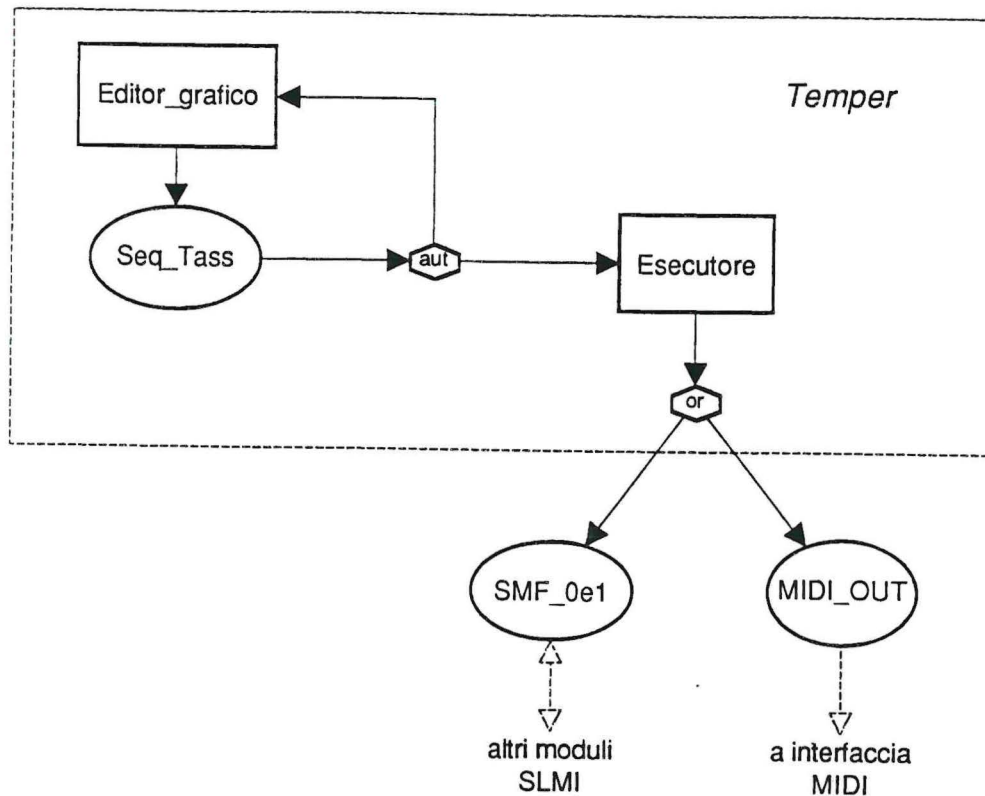
- *Parametri di voci*, che rendono possibile scegliere quali lati devono partecipare alla generazione del suono; a ciascuno di essi si può quindi assegnare una voce (ovvero timbro) distinta, ottenendo interessanti risultati politimbrici e polifonici. Ad esempio si può assegnare ad un lato una voce con funzione melodica e ad un altro una a cui viene affidata la base ritmica, oppure ottenere effetti particolari (ad es. voci che si inseguono o si fondono fra loro). Cambiando il timbro utilizzato si può giungere a rivoluzionare la traduzione in modo radicale.

La generazione del suono è realizzata mediante dispositivi MIDI. *TEMPER*, inoltre, può anche dirigere il suo output musicale su Standard MIDI Files.

Nell'ambito della SLMI, il modulo *TEMPER* si colloca al Livello S/W 2 (Simbolico/Strutturale) (Camurri et al., 1989, parte I, §P1) e per quanto riguarda gli ambiti operativi specifici della SLMI, *TEMPER* potrà essere utilizzato come strumento di ausilio all'attività di realizzazione di *Performance audiovisuali* (Camurri et al., 1989, parte I, §P2).

P6.2 Specifiche funzionali di alto livello.

Diagramma funzionale.



Descrizione funzionale in pseudocodice ADA-like.

-- Descrizione modulo *Temper*

```
package Temper is
    procedure Editor_grafico;
    procedure Esecutore;
end Temper;
```

-- Tipi di strutture dati utilizzati dal modulo *Temper*

```
type Seq_Tass;
--> Sequenze di tassellazioni
type SMF_0e1;
--> Standard MIDI File formato 0 e 1
type MIDI_OUT;
--> Dati standard MIDI diretti a interfaccia MIDI
```

Per quanto riguarda i dettagli implementativi e le modalità d'uso del modulo *Temper* si faccia riferimento al corrispondente documento operativo PFI-2 (M/21, 1991).

P7. Modulo "ScoreSegmenter".

P7.1 Generalità.

Uno dei temi più affascinanti in campo musicale, affrontabili mediante l'uso di strumenti informatici, è quello della strumentazione automatica di brani musicali. Il primo passo da compiere in questa direzione è rappresentato dalla possibilità di *segmentare* brani musicali, cioè di cercare e riconoscere i diversi *oggetti musicali* di cui è composto il brano; dove per *oggetti musicali* si intendono in questo caso quei fraseggi o pensieri musicali che l'autore ha espresso e che ha ripreso e trasformato, secondo i vari canoni musicali, in relazione al periodo storico e alla forma compositiva.

Il modulo *ScoreSegmenter* (Lonati, 1990) della SLMi rappresenta il primo tentativo di realizzare uno strumento informatico che renda possibile accostarsi ai testi musicali non come mera sequenza di note, cioè di eventi sonori, come la pratica strumentale richiede, bensì vedendo i testi stessi come un edificio espressivo fatto da alcuni "mattoncini" composti ed armonizzati fra loro in vario modo.

Il modulo *ScoreSegmenter* legge file generati dall'applicazione commerciale Professional Composer, un programma per l'editoria musicale della Mark of the Unicorn, per ricavarne tutte le informazioni usualmente contenute in uno spartito musicale, e quindi genera file di lavoro in un formato interno. I file di lavoro non contengono più le informazioni secondarie (testi verbali, dinamiche varie, ecc.), mentre contengono in aggiunta tutti gli attributi che la struttura stessa della notazione musicale tradizionale contiene implicitamente, quali la metrica o accentuazione delle singole note in relazione al loro valore ed al contesto in cui sono poste.

Le attività principali che *ScoreSegmenter* consente di effettuare sono: ricerche di occorrenze di oggetti musicali; segmentazione di brani musicali attraverso una analisi del contenuto tematico ed una ricerca delle occorrenze degli elementi tematici individuati.

Nel definire gli algoritmi per la ricerca all'interno del brano delle occorrenze degli oggetti musicali, o di loro sottoparti (entrambi possono essere individuati dall'elaboratore stesso o forniti dall'utente), sono stati considerati due elementi: gli *attributi* della singola nota e la *trasformazione musicale* che la nota stessa ha subito assieme alle note che la precedono e/o seguono. Gli *attributi* sono la durata, l'accento, il nome della nota, l'altezza in semitoni e l'andamento dell'intervallo. Per quanto riguarda le *trasformazioni*, sono state considerate quelle applicate agli attributi di posizione, o grado, sia nella scala tonale che nella scala diatonica. Tali trasformazioni sono state realizzate attraverso tre tipi diversi di operatori algebrici e loro combinazioni; precisamente si tratta dell'operatore di trasposizione, di inversione speculare e di retrogradazione. Tali operatori applicati, come detto, al nome delle note, realizzano le corrispondenti trasformazioni musicali tonali; mentre se applicati all'altezza in semitoni, realizzano le trasformazioni reali. Per fornire buona duttilità a questo strumento di analisi, si è reso possibile intervenire, in mondo interattivo, per variare notevolmente lo stile con cui le ricerche sono condotte, scegliendo quali attributi considerare, quali trasformazioni e quale tasso di variabilità applicare alle analisi tonali.

Per quanto riguarda la segmentazione, ci si è basati sulla forma musicale della fuga, generalizzandola poi per le altre forme. Per la individuazione degli oggetti ci si è basati non solo sulle loro ripetizioni, che sono evidentemente un aspetto necessario ma non sufficiente ai fini musicali. Un tema ha infatti anche un impianto tonale (ad eccezione della musica moderna) e metrico dai quali non è possibile prescindere. Per tener conto anche di questi aspetti vengono formulate delle richieste per le note degli ipotetici temi, in relazione all'appartenenza e affermazione della tonalità a seconda del tipo di oggetto (tetico, acefalo e anacrusico) e in relazione alla esaustività e compiutezza del pensiero musicale, quale effetto della metrica. Anche per questa fase di individuazione dei temi si è predisposta una serie di parametri da specificare interattivamente, sia riguardo gli elementi da valutare nella definizione del tema (tonalità, metrica, lunghezza del fraseggio, numero minimo di ripetizioni, ecc.), che la ripetizione stessa degli oggetti.

Le rilevazioni statistiche risultanti dalle attività di ricerca e segmentazione vengono memorizzate in file di tipo testo, in modo da poterle consultare in tempo differito mediante una qualsiasi applicazione di text editing o word processing.

Nell'ambito della SLMI, il modulo *ScoreSegmenter* si colloca al Livello S/W 2 (Simbolico/Strutturale) (Camurri et al., 1989, parte I, §P1) e per quanto riguarda gli ambiti operativi specifici della SLMI, *ScoreSegmenter* potrà essere utilizzato come strumento di ausilio alle attività di *Analisi/sintesi testi musicali* e *Analisi strutture testi musicali* (Camurri et al., 1989, parte I, §P2).

P7.2 Specifiche funzionali di alto livello.

Diagramma funzionale.

Ved. pag. 24.

Descrizione funzionale in pseudocodice ADA-like.

```
-- Descrizione modulo ScoreSegmenter

package ScoreSegmenter is
    procedure Traduttore;
    procedure Segmentazione;
    procedure Ricerca_sequenzedate;
    procedure Editor_Risultati.txt;
end ScoreSegmenter;

-- Tipi di strutture dati utilizzati dal modulo ScoreSegmenter
type Partiture_PersComp;
--> Partiture codificate nel formato Personal Composer
type FDL;
--> File Di Lavoro per le procedure di analisi
type FDR;
--> File Di Ricerca per la procedura di ricerca sequenze date
type Risultati.txt;
--> File di tipo testo contenente i risultati delle analisi
--> effettuate sulla partitura
type SMF_0e1;
--> Standard MIDI File formato 0 e 1
```


Per quanto riguarda i dettagli implementativi e le modalità d'uso del modulo *ScoreSegmenter* si faccia riferimento al corrispondente documento operativo PFI-2 (M/18, 1991).

P8. Modulo "SampleBuster".

P8.1 Generalità.

La crescente disponibilità di modelli timbrici e di campioni sonori se da una parte arricchisce notevolmente le possibilità di scelta per l'orchestrazione di brani musicali, dall'altra pone il problema di rendere visibili nel modo più rapido e opportuno i timbri. L'esistenza di una ricca banca timbrica di campioni sonori all'interno della SLMi (Camurri et al., 1989, parte I, §P3) ha reso necessario la definizione di un modulo per la gestione delle attività di consultazione di tale banca. Il modulo *SampleBuster* (Penocchio, 1990) è stato concepito non come semplice strumento di accesso ad una banca dati sonora, bensì come un vero e proprio strumento di ausilio all'orchestrazione, sfruttando al meglio le potenzialità di uno strumento ipertestuale quale è HyperCard nell'ambiente Macintosh.

Il modulo *SampleBuster* è costituito da un insieme di stack interconnessi, ciascuno dei quali sviluppa una ben determinata funzione:

- Stack *Home*. Permette di accedere in modo selettivo alle diverse funzionalità di *SampleBuster*.
- Stack *CD catalog*. Gestisce le attività di ricerca selettiva e condizionata di campioni sonori memorizzati su CD-DA.
- Stack *CD Editor*. E' uno stack di servizio per le attività dello stack *CD catalog*.
- Stack *disk 3.5*. Gestisce le attività di ricerca selettiva e condizionata di campioni sonori memorizzati su floppy disk.
- Stack *sampler*. Funge da interfaccia tra i due diversi mondi "samplers" e "CD" e come nucleo operativo per le attività di acquisizione di informazioni sui campionatori.
- Stack *schede S950*. Contiene le informazioni utili per acquisire una conoscenza completa del campionatore AKAI S950.

- Stack *schede S1000*. Contiene le informazioni utili per acquisire una conoscenza completa del campionatore AKAI S1000.
- Stack *orchestrazione*. La funzione di questo stack è di fornire, unitamente allo stack *orchestra*, uno strumento di lavoro a coloro che non sono particolarmente a conoscenza dei problemi relativi all'utilizzazione degli strumenti musicali nel contesto orchestrale.
- Stack *orchestra*. Contiene tutte le informazioni relative agli strumenti musicali, utili per una loro collocazione in un ambito orchestrale.
- Stack *glossario*. E' una raccolta ordinata di spiegazioni dei termini specialistici contenuti nei vari stack.

Nell'ambito della SLMI, il modulo *ScampleBuster* si colloca al Livello S/W 2 (Simbolico/Strutturale) (Camurri et al., 1989, parte I, §P1) e per quanto riguarda gli ambiti operativi specifici della SLMI, *SampleBuster* potrà essere utilizzato come strumento di ausilio all'attività di *Orchestrazione* (Camurri et al., 1989, parte I, §P2).

P8.2 Specifiche funzionali di alto livello.

Diagramma funzionale.

Ved. pag. 27.

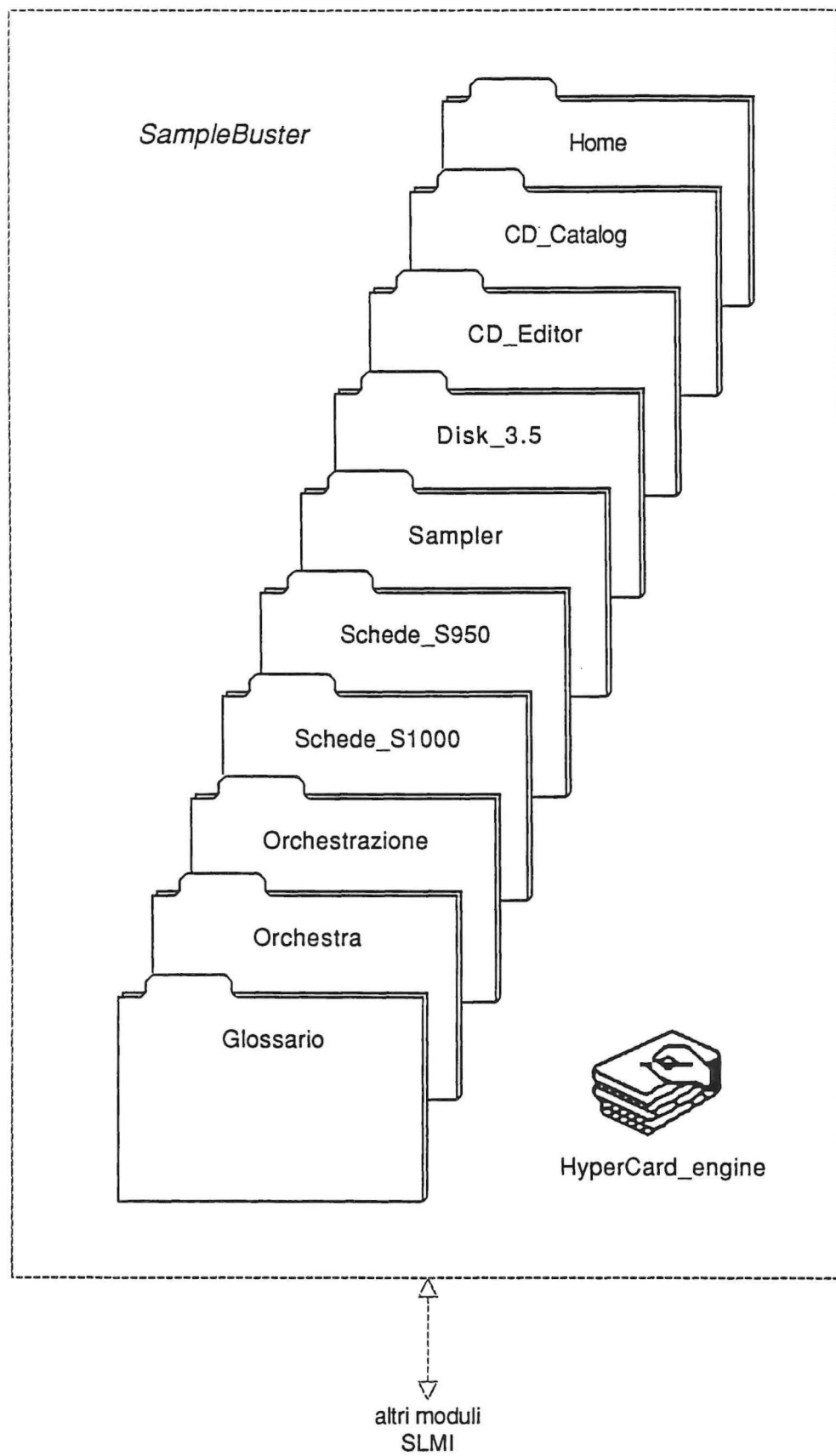
Descrizione funzionale in pseudocodice ADA-like.

```
-- Descrizione modulo SampleBuster

package SampleBuster is
    procedure HyperCard_engine;
end SampleBuster;

-- Tipi di strutture dati utilizzati dal modulo SampleBuster

type Home is HYPERCARD_STACK;
--> Stack per la selezione e l'accesso alle varie funzionalità
```



```

type CD_Catalog is HYPERCARD_STACK;
--> Stack per la ricerca selettiva di campioni sonori
--> memorizzati su CD-DA
type CD_Editor is HYPERCARD_STACK;
--> Stack di servizio per lo stack CD_Catalog
type Disk_3.5 is HYPERCARD_STACK;
--> Stack per la ricerca selettiva di campioni sonori
--> memorizzati su floppy disk
type Sampler is HYPERCARD_STACK;
--> Stack di collegamento tra mondo "sampler" e mondo "CD-DA"
type Schede_S950 is HYPERCARD_STACK;
--> Stack relativo al campionatore AKAI S950
type Schede_S1000 is HYPERCARD_STACK;
--> Stack relativo al campionatore AKAI S1000
type Orchestrazione is HYPERCARD_STACK;
--> Stack di ausilio all'orchestrazione di brani musicali
type Orchestra is HYPERCARD_STACK;
--> Stack di informazioni relative agli strumenti musicali
type Glossario is HYPERCARD_STACK;
--> Stack glossario di termini specialistici a cui si fa
--> riferimento nel modulo SampleBuster

```

Per quanto riguarda i dettagli implementativi e le modalità d'uso del modulo *SampleBuster* si faccia riferimento al corrispondente documento operativo PFI-2 (M/13, 1991).

P9. Modulo "Libreria di routines di lettura/scrittura di Standard MIDI File 1.0".

P9.1 Generalità.

La definizione di uno standard per la codifica dei dati MIDI (IMA, 1988), ha consentito di fissare una valida modalità di scambio di dati MIDI fra programmi diversi, prescindendo dai computer sui quali tali programmi vengono eseguiti, e soprattutto dalle singole esigenze degli autori dei programmi stessi. Grazie agli SMF, si è implicitamente definito un linguaggio musicale comune per i personal computer che ha permesso di superare le barriere altrimenti esistenti tra programmi musicali diversi. Anche nell'ambito della SLMI è sorto il problema di far

comunicare i diversi moduli, per poter lavorare sugli stessi archivi in aree operative differenti. Di conseguenza è stato realizzato un modulo di supporto allo sviluppo di tutti gli altri moduli, per quanto riguarda la lettura e la scrittura di SMF.

Il modulo *Libreria di routines di lettura/scrittura di Standard MIDI File 1.0* (Bettitoni, 1990) è costituito da routines scritte in PASCAL, utilizzabili in ambiente MPW (Macintosh Programmer's Workshop), per la scrittura/codifica e per la lettura/decodifica di Standard MIDI Files (versione 1.0) nei formati 0 e 1, secondo le specifiche fornite dalla *International MIDI Association* (IMA, 1988). In particolare comprende routines per la codifica e la decodifica di numeri espressi in forma di "variable-length quantity" (rappresentazione "compressa" di valori numerici utilizzata negli SMF), routines per la lettura e la scrittura di header generali e header di traccia, routines di lettura/decodifica e codifica/scrittura di metaeventi, routines di lettura/decodifica e codifica/scrittura di eventi, routines di supporto per la diagnostica.

Nell'ambito della SLMI, il modulo *Libreria di routines di lettura/scrittura di Standard MIDI File 1.0* è stato utilizzato, in aggiunta al software di base, nello sviluppo di tutti i moduli sperimentali del Livello S/W 2 (Simbolico/Strutturale) (Camurri et al., 1989, parte I, §P1).

P9.2 Specifiche funzionali di alto livello.

Diagramma funzionale.

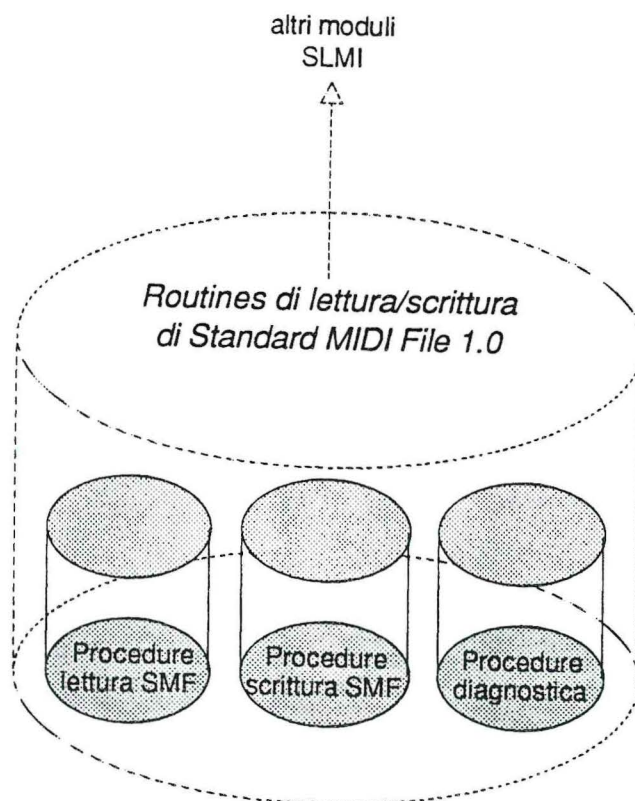
Ved. pag. 29.

Descrizione funzionale in pseudocodice ADA-like.

```
-- Descrizione modulo Routines di lettura/scrittura di Standard  
-- MIDI File 1.0
```

```
package Routines_R&W_SMF is  
--> procedure per la lettura di SMF 1.0;  
--> procedure per la scrittura di SMF 1.0;  
--> procedure di supporto alla diagnostica;  
end Routines_R&W_SMF;
```

Per quanto riguarda i dettagli implementativi e le modalità d'uso del modulo *Routines di lettura/scrittura di Standard MIDI File 1.0* si faccia riferimento al corrispondente documento operativo PFI-2 (M/14, 1991).



P10. Modulo "Ambiente sperimentale per microprogrammi DSP per Sound Accelerator Card".

P10.1 Generalità.

Il modulo *Ambiente sperimentale per microprogrammi DSP per Sound Accelerator Card* (Casali, 1990) consente di sperimentare tre tecniche di sintesi (ved. anche §P4): sintesi mediante Funzioni di Due Variabili (TVF); sintesi Waveshaping (WSH); sintesi mediante l'uso di Wavelet (WLT). Il modulo è dotato di un'adeguata interfaccia utente e fa uso diretto del modulo *Microprogrammi per Sequencer MIDI/DSP e Sound Accelerator Card* (ved. §P4).

L'interfaccia utente è studiata in modo da potersi adeguare strutturalmente in funzione dell'integrazione con il modulo *Sequencer MIDI/DSP* (ved. §P3).

Nell'ambito della SLMI, il modulo *Ambiente sperimentale per microprogrammi DSP per Sound Accelerator Card* si colloca al Livello S/W 1 (Esecutivo/Operativo) (Camurri et al., 1989, parte I, §P1) e per quanto riguarda gli ambiti operativi specifici della SLMI, potrà essere utilizzato come strumento di ausilio alle attività di *Analisi/sintesi del suono* e di *Preparazione di modelli timbrici* (Camurri et al., 1989, parte I, §P2).

P10.2 Specifiche funzionali di alto livello.

Diagramma funzionale.

Ved. pag. 32.

Descrizione funzionale in pseudocodice ADA-like.

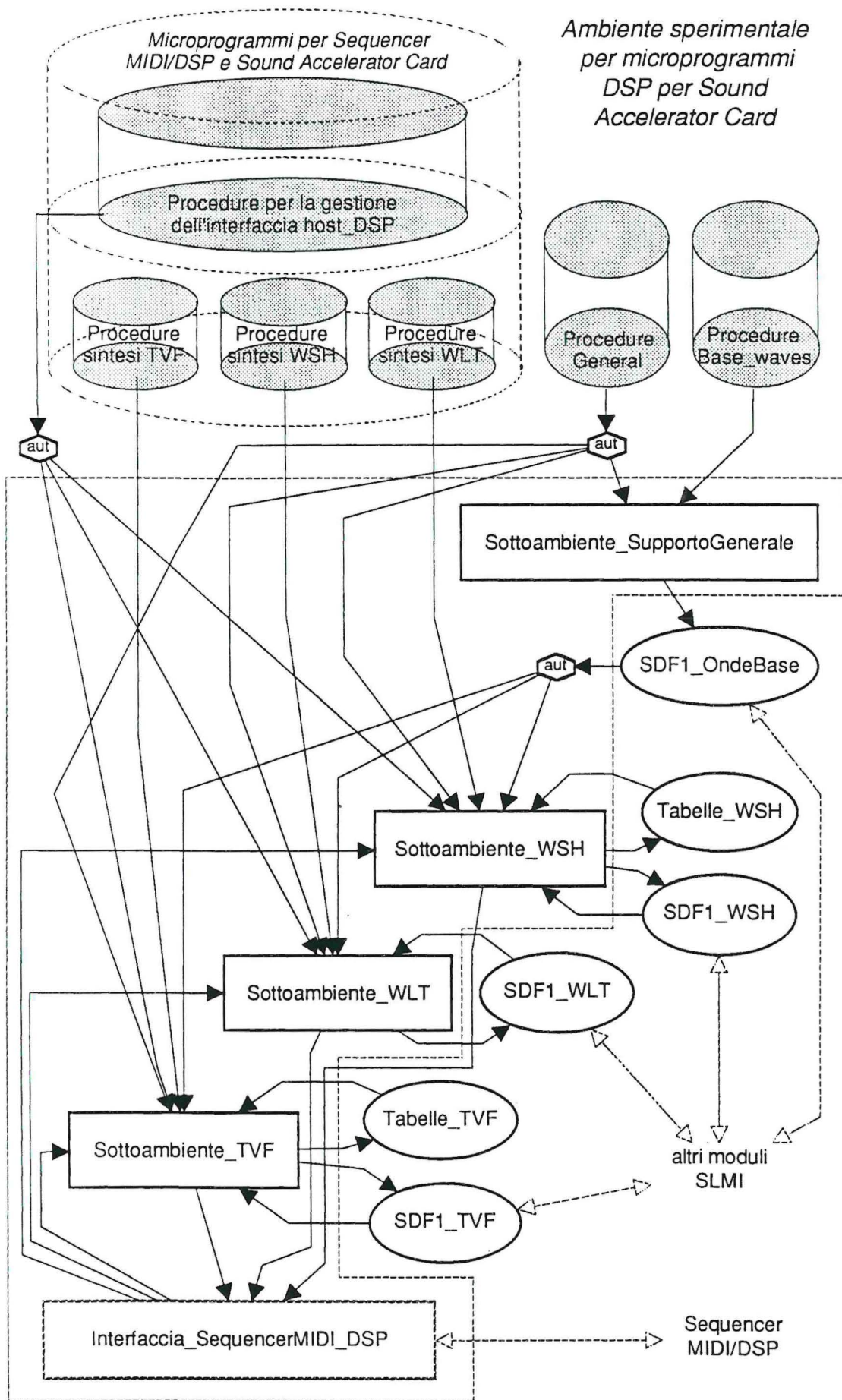
```
-- Descrizione modulo Ambiente sperimentale per microprogrammi
-- DSP per Sound Accelerator Card

package AmbienteSperimentaleDSP is
    procedure Sottoambiente_SupportoGenerale;
    procedure Sottoambiente_TVF;
    procedure Sottoambiente_WSH;
    procedure Sottoambiente_WLT;
    procedure Interfaccia_SequencerMIDI_DSP;
end AmbienteSperimentaleDSP;

package General is
--> procedure e funzioni di utilità (di uso generale)
end General;

package Base_waves is
--> procedure e funzioni di uso generale per la costruzione
--> di forme d'onda base
end Base_waves;

package Microprogrammi_DSP is
--> ved. §P4.2
end Microprogrammi_DSP;
```



```

-- Tipi di strutture dati utilizzati dal modulo
-- Ambiente sperimentale per microprogrammi DSP per
-- Sound Accelerator Card

type SDF1_OndeBase;
--> SoundFile formato 1 (standard Digidesign) contenenti
--> forme d'onda base
type SDF1_TVF;
--> SoundFile formato 1 (standard Digidesign) contenenti
--> forme d'onda per la sintesi TVF
type SDF1_WSH;
--> SoundFile formato 1 (standard Digidesign) contenenti
--> forme d'onda per la sintesi WSH
type SDF1_WLT;
--> SoundFile formato 1 (standard Digidesign) contenenti
--> forme d'onda per la sintesi WLT
type Tabelle_TVF;
--> Tabelle di valori parametrici per la sintesi TVF
type Tabelle_WSH;
--> Tabelle di valori parametrici per la sintesi WSH

```

Per quanto riguarda i dettagli implementativi e le modalità d'uso del modulo *Ambiente sperimentale per microprogrammi DSP per Sound Accelerator Card* si faccia riferimento al corrispondente documento operativo PFI-2 (M/15, 1991).

P11. Interfaccia utente ipertestuale.

P11.1 Generalità.

La funzione principale del modulo *Interfaccia utente ipertestuale* è di rendere visibile l'insieme degli ambiti operativi disponibili nella SLMI a ogni possibile classe di utenti, e di rendere disponibili, all'interno dell'ambito operativo prescelto, i moduli sperimentali e commerciali della SLMI che consentono di svolgere le attività relative a quell'ambito.

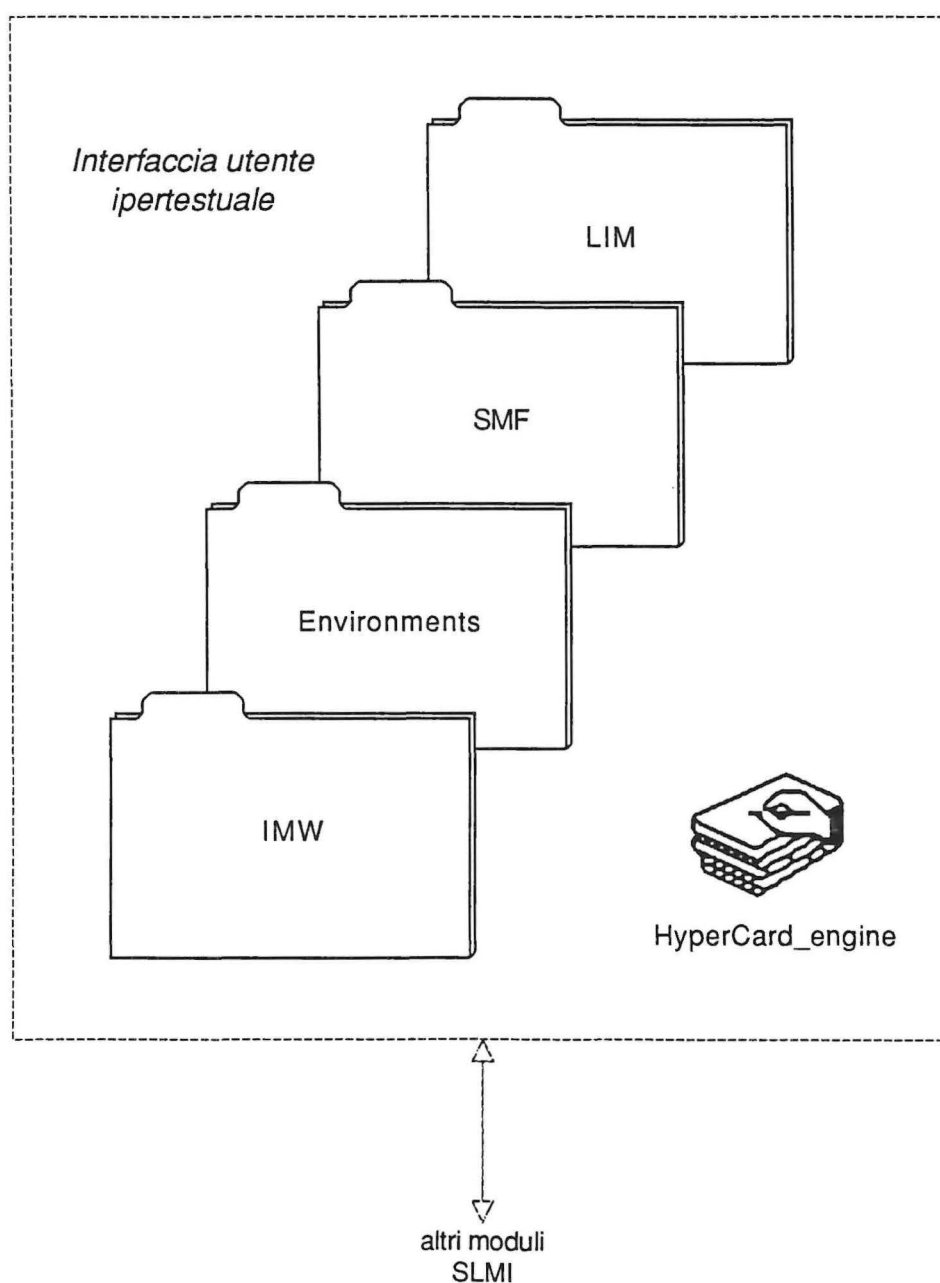
L'*interfaccia utente ipertestuale* è realizzata in ambiente Hypercard, ed è strutturata come un insieme di 4 stack interconnessi:

- Stack *imw*. E' lo stack di apertura dell'interfaccia. In esso, la prima scheda è una presentazione dei moduli sperimentali e delle applicazioni commerciali. I pulsanti associati ai moduli sperimentali conducono alle schede dello stack LIM, contenenti descrizioni sintetiche relative ai moduli stessi. Mentre la seconda e la terza scheda presentano in dettaglio le varie applicazioni commerciali raggruppate in classi. L'utente ha la possibilità di aggiungere in ogni lista (classe) i moduli e/o le applicazioni che ritiene opportuno utilizzare per il suo lavoro, oltre che di rimuovere quelle che non ritiene più utili. Infine l'ultima scheda presenta l'insieme degli ambiti operativi in cui è possibile lavorare: ogni utente ha la possibilità di operare nell'ambiente che gli è più congeniale.
- Stack *environments*. Dall'ultima scheda dello stack *imw* si accede alle schede di questo stack, una per ciascun ambito operativo scelto (per es. *Composizione* o *Performance*). Le schede hanno tutte la stessa impostazione grafica; cambia l'insieme delle applicazioni e/o dei moduli a cui l'utente può accedere per ogni ambiente. In tutte le schede sono sempre presenti applicazioni di utilità per l'ambiente Macintosh come MacWrite e ResEdit. I pulsanti collegati alle classi di applicazioni commerciali rimandano alle due schede dello stack *imw* con le liste di tali applicazioni; quelli collegati ai moduli sperimentali aprono direttamente i moduli stessi. Un pulsante conduce ad un catalogo di SMF (Stack SMF). Mentre un altro rimanda all'ultima scheda dello stack *imw*, dalla quale l'utente può scegliere di operare in un altro ambiente accedendo quindi ad un'altra scheda di questo stack.
- Stack SMF. Poichè tutti i moduli sperimentali e le applicazioni commerciali che compongono la SLMi possono scambiarsi informazioni MIDI sotto forma di Standard MIDI Files v.1.0, si è pensato di realizzare un catalogo di SMF gestibile dall'utente. Ogni scheda di questo stack contiene le informazioni essenziali su un file MIDI che l'utente ha ritenuto opportuno catalogare. Le informazioni sono immesse da tastiera ogni volta che l'utente si crea una nuova scheda per un nuovo file. C'è inoltre la possibilità di ascoltare in modalità "preascolto" gli Standard MIDI Files presenti nel catalogo, senza quindi essere costretti ad attivare i moduli o le applicazioni che li hanno generati.
- Stack LIM. Le schede di questo stack contengono descrizioni sintetiche dei moduli sperimentali realizzati al LIM e sono accessibili tramite i pulsanti (presenti nella scheda di apertura dello stack *imw*) con i nomi e le icone dei moduli stessi. Inoltre un pulsante renderà possibile l'accesso ai manuali operativi di ciascun modulo sperimentale.

Nell'ambito della SLMI, il modulo *Interfaccia utente ipertestuale* si colloca ad un Livello S/W specifico per l'interfaccia utente (ved. rapporto PFI-2 R2, parte I, §P1).

P11.2 Specifiche funzionali di alto livello.

Diagramma funzionale.



Descrizione funzionale in pseudocodice ADA-like.

```
-- Descrizione modulo Interfaccia utente ipertestuale

package InterfacciaSLMI is
    procedure HyperCard_engine;
end InterfacciaSLMI;

-- Tipi di strutture dati utilizzati dal modulo Interfaccia
-- utente ipertestuale

type IMW is HYPERCARD_STACK;
--> Stack per la selezione e l'accesso alle varie funzionalità
type Environments is HYPERCARD_STACK;
--> Stack ambienti operativi della SLMI
type SMF is HYPERCARD_STACK;
--> Stack gestione catalogo SMF
type LIM is HYPERCARD_STACK;
--> Stack di accesso alla documentazione sui moduli della SLMI
```

Per quanto riguarda i dettagli implementativi e le modalità d'uso del modulo *Interfaccia utente ipertestuale* si faccia riferimento al corrispondente documento operativo PFI-2 (M/11, 1991).

P12. Modulo "LASy Workbench".

P12.1 Generalità.

LASy (Linear Automata Synthesis) è una nuova tecnica di sintesi che utilizza automi cellulari per generare forme d'onde in continua evoluzione (Chareyron, 1980a e 1980b).

Il modulo *LASy Workbench* (Chareyron, 1990) implementa la sintesi sonora LASy su un Apple Macintosh II dotato di scheda Sound Accelerator della Digidesign.

Il programma offre in una architettura unica, sia un *ambiente di ricerca* e di costruzione di nuovi suoni, sia un *ambiente esecutivo* nel quale sfruttare musicalmente il prodotto della sintesi. Dal punto di vista dell'utente non esiste separazione tra i due ambienti: i suoni creati con gli editor a disposizione

possono essere subito suonati e controllati in tempo reale usando la tastiera del computer, il mouse e/o una tastiera musicale collegata via MIDI. *LASy Workbench* può essere considerato come la realizzazione software di uno strumento musicale LASy dotato di avanzate possibilità di editing.

Ambiente di ricerca. La generazione di un suono mediante sintesi LASy richiede la presenza di due tabelle numeriche e la scelta di un insieme di parametri. L'ambiente di ricerca prevede a questo scopo diversi moduli collegati:

- * Il modulo di costruzione degli strumenti, nel quale l'utente sceglie gli elementi del suono appena citati e precisa l'effetto dei controlli in tempo reale.

- * I moduli di creazione dei due tipi di tabelle richieste (forma d'onda e regola di transizione) che offrono una famiglia di algoritmi parametrizzabili per generare il contenuto di queste tabelle.

- * I moduli di editing delle tabelle. Sono previsti sia un editor alfanumerico sia un editor grafico.

- * Il modulo di archiviazione che gestisce i differenti tipi di file (forme d'onda, regole di transizione, strumenti) usati dal programma. Questo modulo prevede anche la produzione di file utilizzabili dal modulo di sintesi LASy integrato nel *Sequencer MIDI-DSP* (ved. §P3).

Ambiente di esecuzione. La generazione sonora viene controllata in tre diversi modi (non esclusivi): usando solo i dispositivi di controllo (tastiera alfanumerica e mouse), usando una tastiera musicale (o un sequencer) collegata via MIDI, oppure direttamente grazie a un modulo software. I moduli richiesti sono i seguenti:

- * Il modulo di sintesi, presente sotto forma di microprogramma da caricare nella scheda DSP.

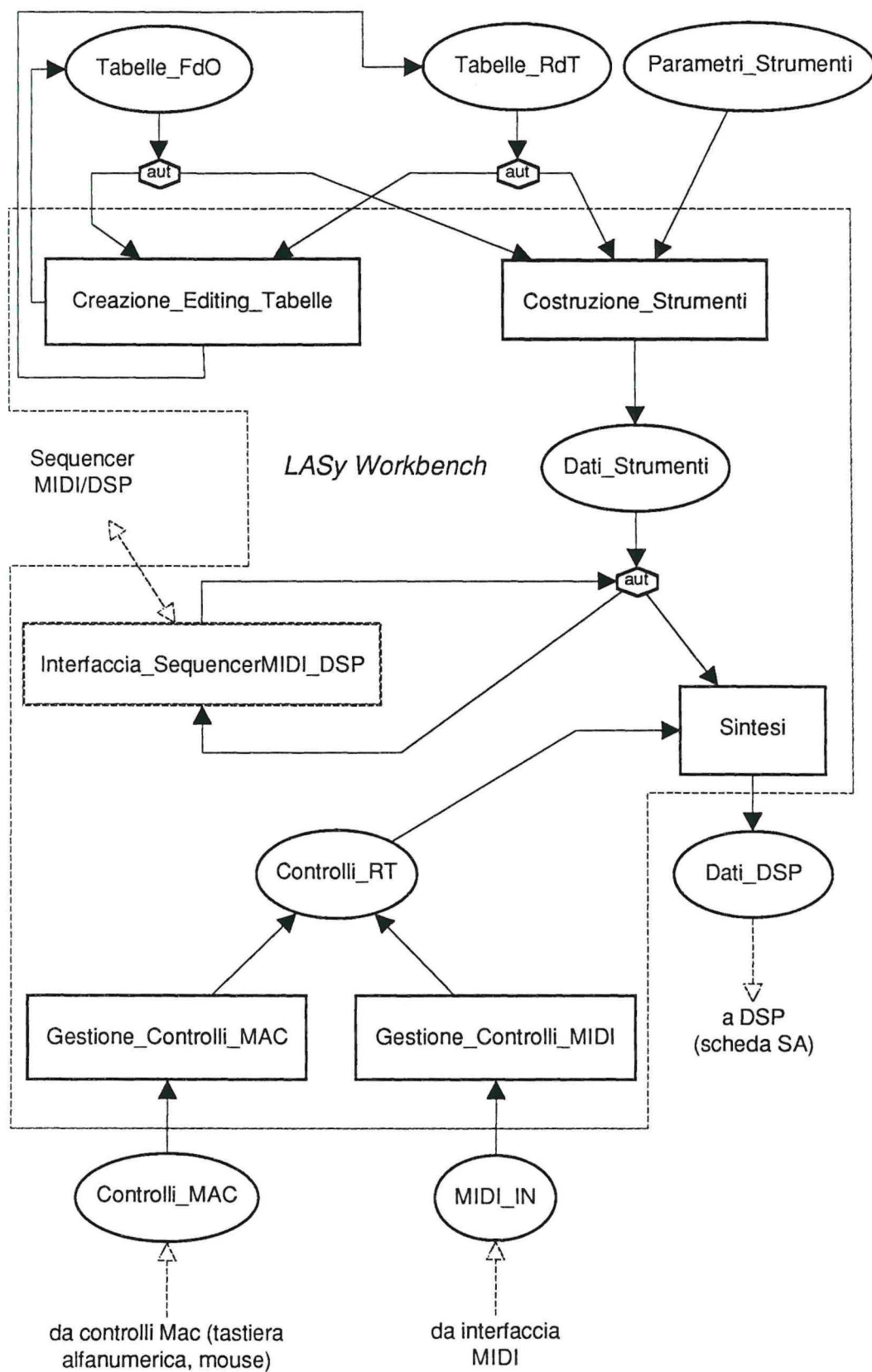
- * Il modulo MIDI dedicato all'interpretazione dei messaggi presenti sulla porta seriale del calcolatore.

- * Il modulo di controllo in tempo reale del suono. L'evoluzione del suono generato viene gestita in tempo reale dall'interprete a seconda della sua azione su mouse, tastiera o MIDI controllers.

Nell'ambito della SLMi, il modulo *LASy Workbench* si colloca al Livello S/W 1 (Esecutivo/Operativo) (Camurri et al., 1989, parte I, §P1) e per quanto riguarda gli ambiti operativi specifici della SLMi, potrà essere utilizzato come strumento di ausilio alle attività di *Analisi/sintesi del suono* e di *Preparazione di modelli timbrici* (Camurri et al., 1989, parte I, §P2).

P12.2 Specifiche funzionali di alto livello.

Diagramma funzionale.



Descrizione funzionale in pseudocodice ADA-like.

```
-- Descrizione modulo LASy Workbench

package LASyWorkbench is
    procedure Creazione_Editing_Tabelle;
    procedure Costruzione_Strumenti;
    procedure Sintesi;
    procedure Gestione_Controlli_MIDI;
    procedure Gestione_Controlli_MAC;
    procedure Interfaccia_SequencerMIDI_DSP;
end LASyWorkbench;

-- Tipi di strutture dati utilizzati dal modulo LASy Workbench

type Tabelle_FdO;
--> Tabelle Forme D'Onda
type Tabelle_RdT;
--> Tabelle Regole Di Transizione
type Parametri_Strumenti;
--> Parametri per la costruzione degli strumenti
type Dati_Strumenti;
--> Dati per la sintesi degli strumenti
type Dati_DSP;
--> Dati da inviare al DSP (scheda Sound Accelerator)
type MIDI_IN;
--> Dati standard MIDI provenienti da interfaccia MIDI
type Controlli_RT;
--> Dati per il controllo in tempo reale
type Controlli_MAC;
--> Dati provenienti dai dispositivi di controllo (tastiera
--> alfanumerica, mouse) del Macintosh
```

Per quanto riguarda i dettagli implementativi e le modalità d'uso del modulo *LASy Workbench* si faccia riferimento al corrispondente documento operativo PFI-2 (M/25, 1991).

P13. Modulo "Traslitteratore".

P13.1 Generalità.

Il modulo *Traslitteratore* (Bastardi, 1987) (Crippa, 1987) consente di realizzare in modo automatico, secondo alcune regole di traslitterazione, la trasformazione di un testo letterario in uno spartito musicale.

Le analogie che consentono di trasformare un testo in musica sono sia a livello sonoro (fonemi, vocali, consonanti, sillabe, silenzi, ecc.), che a livello di struttura propria del testo (un sonetto è ricco di rime che sono riproposte con cadenze ben precise, oppure un dialogo è ricco di coppie di frasi domanda-risposta in cui la risposta riprende tutta o in parte la domanda); l'identificazione di queste analogie consente di riprodurre nel brano musicale una struttura di eventi musicali simile alla struttura del testo originale.

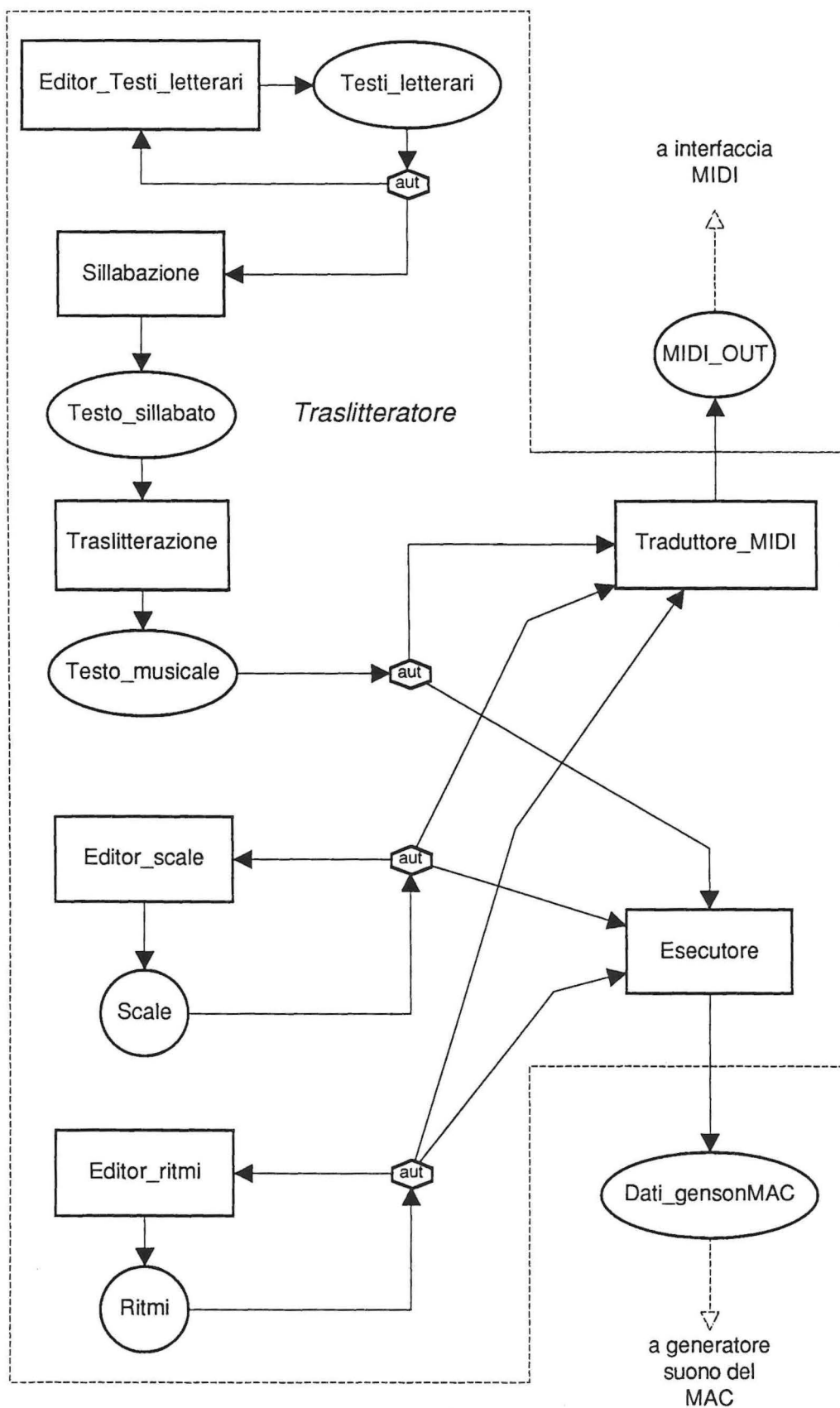
Inoltre alcune caratteristiche del testo, come la punteggiatura, i dittonghi, gli iati, la lunghezza della sillaba, ecc., influiscono direttamente sulla generazione di altrettante caratteristiche musicali, come ornamenti (acciaccature, mordenti), durate, dinamica, pause, articolazioni (accelerando, ritardando), ecc.

Il modulo consente anche di ascoltare i brani ottenuti dalla traslitterazione, sia attraverso i suoni prodotti con il sintetizzatore interno al Macintosh, oppure utilizzando uno o più strumenti musicali esterni collegati all'elaboratore mediante interfaccia MIDI. Questa seconda alternativa è realizzata con un traduttore che permette di codificare i dati secondo lo standard MIDI. Tale traduttore è in grado di tradurre anche i simboli di accelerando e ritardando che si possono inserire nel testo, ottenendo così un qualcosa di innovativo nel campo dell'esecuzione di partiture con elaboratore: in questo modo è infatti possibile rimuovere, in parte, quella sensazione di appiattimento espressivo caratteristico di molti programmi per la realizzazione e l'esecuzione di musica con l'elaboratore.

Nell'ambito della SLMI, il modulo *Traslitteratore* si colloca al Livello S/W 2 (Simbolico/Strutturale) (Camurri et al., 1989, parte I, §P1) e per quanto riguarda gli ambiti operativi specifici della SLMI, potrà essere utilizzato come strumento di ausilio alle attività di *Composizione*, *Sintesi di testi musicali* e *Performance musicale* (Camurri et al., 1989, parte I, §P2).

P13.2 Specifiche funzionali di alto livello.

Diagramma funzionale.



Descrizione funzionale in pseudocodice ADA-like.

```
-- Descrizione modulo Traslitteratore

package Traslitteratore is
    procedure Editor_Testi_letterari;
    procedure Editor_scale;
    procedure Editor_ritmi;
    procedure Sillabazione;
    procedure Traslitterazione;
    procedure Traduttore_MIDI;
    procedure Esecutore;
end Traslitteratore;

-- Tipi di strutture dati utilizzati dal modulo Traslitteratore

type Testi_letterari;
--> Files di tipo testo contenenti i testi letterari
type Scale;
--> Scale musicali da utilizzare in fase di esecuzione
type Ritmi;
--> Ritmi musicali da utilizzare in fase di esecuzione
type Testo_sillabato;
--> Testo letterario suddiviso in sillabe e pronto per la
--> fase di traslitterazione
type Testo_musicale;
--> Dati musicali ottenuti dalla traslitterazione
--> del testo letterario
type MIDI_OUT;
--> Dati standard MIDI diretti a interfaccia MIDI
type Dati_gensonMAC;
--> Dati per il generatore di suoni del Macintosh
```

Per quanto riguarda i dettagli implementativi e le modalità d'uso del modulo *Traslitteratore* si faccia riferimento al corrispondente documento operativo PFI-2 (M/20, 1991).

P14. Riferimenti bibliografici.

- Ballista, A. (1990). Un sequencer per il controllo integrato di unità di sintesi MIDI e unità Digital Signal Processing. T. di L. in Scienze dell'Informazione, Università degli Studi, Milano.
- Bastardi, M. (1987). Traslitterazione di testi letterari in partiture musicali: strumenti di sintesi. T. di L. in Scienze dell'Informazione, Università degli Studi, Milano.
- Benvestito, M. (1990). Progetto di un sistema per elaborare ed eseguire oggetti musicali e grafico-musicali in tempo reale. T. di L. in Scienze dell'Informazione, Università degli Studi, Bari.
- Bettitoni, P. (1990). Un sistema software per il mixing di processi MIDI. T. di L. in Scienze dell'Informazione, Università degli Studi, Milano.
- Borgonovo, A., Haus, G. (1986). Sound Synthesis by Means of Two-Variable Functions: Experimental Criteria and Results. *Computer Music Journal*, 10(3), 57-71, MIT Press, Cambridge, MA.
- Camurri, A., Haus, G., Zaccaria, R. (1986). Describing and Performing Musical Processes by means of Petri Nets. *Interface*, 15(1), 1-25, Swets & Zeitlinger B.V., Amsterdam.
- Camurri, A., Haus, G., Stiglitz, A., Zaccaria, R., (1989). LRC C4 MUSIC: architettura, specifiche funzionali di alto livello e standard di documentazione della STAZIONE DI LAVORO MUSICALE INTELLIGENTE. PFI-2 CNR, 7/09, 11/89.
- Casali, E. (1990). Una libreria di microprogrammi di sintesi per DSP Motorola 56001. T. di L. in Scienze dell'Informazione, Università degli Studi, Milano.
- Chareyron, J. (1988a). Sound Synthesis and Processing by Means of Linear Cellular Automata. In C.Lischka and J. Fritsch, eds. *Proceedings of the 1988 International Computer Music Conference (Supplement)*.
- Chareyron, J. (1988b). Wavetable come Automa Cellulare: una Nuova Tecnica di Sintesi. *Atti del VII Colloquio di Informatica Musicale*. Roma, Italy: Edizioni Arti Grafiche Ambrosini, pp 174-177.
- Chareyron, J. (1990). Digital Synthesis of Self-Modifying Waveforms by Means of Linear Automata. *Computer Music Journal*, 14(3), M.I.T. Press, Cambridge, MA.
- Crippa, P. (1987). Traslitterazione di testi letterari in partiture musicali: strumenti di analisi. T. di L. in Scienze dell'Informazione, Università degli Studi, Milano.
- Degli Antoni, G., Haus, G. (1983). Music and Causality. *Proc. 1982 International Computer Music Conference*, 279-296, Computer Music Association Publ., San Francisco.
- Dutilleux, P., Grossmann, A., Kronland-Martinet, R. (1988). Application of the Wavelet Transform to the Analysis, Transformation and Synthesis of Musical Sound. *AES 85th Convention Los Angeles* No. 2727 (A-2).
- Haus, G., Morini, P. (1991). TEMPER: a System for the Synthesis of Tessellation Animation and Music. *Leonardo, ISAST, Journal of the ISAST, Pergamon Journals*, San Francisco, CA (submitted).
- Haus, G., Rodriguez, A. (1989). Music Description and Processing by Petri Nets. *1988 Advances on Petri Nets, LNCS*, 340, 175-199, Springer, Berlin.
- Haus, G., Sametti, A. (1991). "SCORESYNTH: a System for the Synthesis of Music Scores based on Petri Nets and a Music Algebra. *IEEE Computer*, (submitted).

Haus, G., Stiglitz, A. (1990). A Software Tool for the Functional Performance of Music. Proc. SISEA (Second International Symposium on Electronic Art), Groningen, NL.

IMA (1988). Standard MIDI Files 1.0. doc. distribuito da The International MIDI Association, Los Angeles, CA.

Lazzini, L. (1989). Un sistema per la generazione di strutture musicali mediante analisi/sintesi di pattern seriali. T. di L. in Scienze dell'Informazione, Università degli Studi, Milano.

Lonati, F. (1990). Metodi, algoritmi e loro implementazione per la segmentazione automatica di partiture musicali. T. di L. in Scienze dell'Informazione, Università degli Studi, Milano.

Mitsubishi, Y. (1982). Audio Signal Synthesis by Functions of Two Variables. JAES, 30 (10), 701-706.

Morini, P. (1987). Interpretazione musicale di divisioni regolari del piano. T. di L. in Scienze dell'Informazione, Università degli Studi, Milano.

Penocchio, G. (1990). Analisi, disegno e realizzazione di un prototipo software per la gestione di timbri musicali basato su strutture ipertestuali. Progetto di Laurea, Ingegneria Elettronica, Politecnico di Milano.

Petri, C.A. (1976). General Net Theory. Proceedings of the Joint IBM & Newcastle upon Tyne Seminar on Computer Systems Design.

Riotte, A. (1963). Génération de séries équilibrées. Relazione interna Euratom n.353, Ispra.

Roads, C. (1979). A tutorial on Non-Linear Distorsion or Waveshaping Synthesis. Computer Music Journal, 3 (2), 29-34, MIT Press, Cambridge, MA.

Sametti, A. (1988). Un sistema per la sintesi di partiture musicali mediante esecuzione di Reti di Petri. T. di L. in Scienze dell'Informazione, Università degli Studi, Milano.

Scolaro, S. (1990). Modelli di partiture musicali mediante Reti di Petri: strumenti software e casi esemplari. T. di L. in Scienze dell'Informazione, Università degli Studi, Milano.

M/03: Guida operativa del modulo "ScoreSynth", Tech. Report PFI2-L.I.M., 1991.

M/08: Guida operativa del modulo "MusSer", Tech. Report PFI2-L.I.M., 1991.

M/05: Guida operativa del modulo "Sequencer MIDI/DSP", Tech. Report PFI2-L.I.M., 1991.

M/10: Note tecniche del modulo "Libreria microprogrammi per Sequencer MIDI-DSP e Sound Accelerator Card (TVF, Automi Cellulari, Waveshaping)", Tech. Report PFI2-L.I.M., 1991.

M/23: Guida operativa del modulo "Functional Performer", Tech. Report PFI2-L.I.M., 1991.

M/21: Guida operativa del modulo "Temper", Tech. Report PFI2-L.I.M., 1991.

M/18: Guida operativa del modulo "ScoreSegmenter ", Tech. Report PFI2-L.I.M., 1991.

M/13: Guida operativa del modulo "SampleBuster", Tech. Report PFI2-L.I.M., 1991.

M/14: Guida operativa del modulo "Routines di lettura/scrittura di Standard MIDI File 1.0", Tech. Report PFI2-L.I.M., 1991.

M/15: Guida operativa del modulo "Ambiente sperimentale per microprogrammi DSP per Sound Accelerator Card (TVF, Automi Cellulari, Waveshaping, Wavelet)", Tech. Report PFI2-L.I.M., 1991.

M/11: Guida operativa dell'interfaccia utente ipertestuale della "Stazione di lavoro musicale intelligente", Tech. Report PFI2-L.I.M., 1991.

M/25: Guida operativa del modulo "LASy Workbench", Tech. Report PFI2-L.I.M., 1991.

M/20: Guida operativa del modulo "Traslitteratore", Tech. Report PFI2-L.I.M., 1991.

