



**CONSIGLIO NAZIONALE  
DELLE RICERCHE**

Isabella Pighi<sup>^</sup> e AA.VV.

**INTEGRAZIONE DELLA  
ARCHITETTURA E DELLE  
SPECIFICHE FUNZIONALI DI ALTO  
LIVELLO DEI MODULI DELLA  
"STAZIONE DI LAVORO MUSICALE  
INTELLIGENTE"**  
**Unità Operativa Univ. degli Studi di Milano**

N. 7/117

2/93

**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

Isabella Pighi<sup>^</sup> e AA.VV.

**INTEGRAZIONE DELLA  
ARCHITETTURA E DELLE  
SPECIFICHE FUNZIONALI DI ALTO  
LIVELLO DEI MODULI DELLA  
"STAZIONE DI LAVORO MUSICALE  
INTELLIGENTE"  
Unità Operativa Univ. degli Studi di Milano**

N. 7/117

2/93

Rapporto Tecnico

<sup>^</sup> LIM - Laboratorio di Informatica Musicale  
Dipartimento di Scienze dell'Informazione  
Università degli Studi di Milano.



## **INDICE:**

### **Premessa generale.**

- P0. Introduzione.**
- P1. Architettura della SLMI.**
- P2. Discussione schematica delle funzionalità della SLMI.**
- P3. Librerie disponibili nella SLMI.**
- P4. Standard utilizzati nell'implementazione.**
- P5. Specifiche funzionali dei moduli della SLMI.**
  - P5.1. Modulo "ScoreSynth".**
    - P5.1.1 Generalità.
    - P5.1.2 Specifiche funzionali di alto livello.
  - P5.2. Modulo "MusSer".**
    - P5.2.1 Generalità.
    - P5.2.2 Specifiche funzionali di alto livello.
  - P5.3. Modulo "Traslitteratore".**
    - P5.3.1 Generalità.
    - P5.3.2 Specifiche funzionali di alto livello.
  - P5.4. Modulo "Functional Performer".**
    - P5.4.1 Generalità.
    - P5.4.2 Specifiche funzionali di alto livello.
  - P5.5. Modulo "TEMPER".**
    - P5.5.1 Generalità.
    - P5.5.2 Specifiche funzionali di alto livello.
  - P5.6. Modulo "ScoreSegmenter".**
    - P5.6.1 Generalità.
    - P5.6.2 Specifiche funzionali di alto livello.
  - P5.7. Modulo "SampleBuster".**
    - P5.7.1 Generalità.
    - P5.7.2 Specifiche funzionali di alto livello.
  - P5.8. Modulo "Sequencer MIDI/DSP".**
    - P5.8.1 Generalità.
    - P5.8.2 Specifiche funzionali di alto livello.
  - P5.9. Modulo "Microprogrammi per Sequencer MIDI/DSP e Sound Accelerator Card".**
    - P5.9.1 Generalità.
    - P5.9.2 Specifiche funzionali di alto livello.



- P5.10. Modulo "Libreria di routine di lettura/scrittura di Standard MIDI File 1.0".**
  - P5.10.1 Generalità.
  - P5.10.2 Specifiche funzionali di alto livello.
- P5.11. Modulo "Ambiente sperimentale per microprogrammi DSP per Sound Accelerator Card".**
  - P5.11.1 Generalità.
  - P5.11.2 Specifiche funzionali di alto livello.
- P5.12. Modulo "LASy Workbench".**
  - P5.12.1 Generalità.
  - P5.12.2 Specifiche funzionali di alto livello.
- P5.13. Modulo "TimbreLab".**
  - P5.13.1 Generalità.
  - P5.13.2 Specifiche funzionali ad alto livello.
- P5.14. Modulo "VoiceLab".**
  - P5.14.1 Generalità.
  - P5.14.2 Specifiche funzionali di alto livello.
- P5.15. Modulo "Cellular Automata Sound Processor".**
  - P5.15.1 Generalità.
  - P5.15.2 Specifiche funzionali di alto livello di CellAut.
  - P5.15.3 Specifiche funzionali di alto livello di FilterAut.
- P5.16. Modulo "Driver MIDI/DSP".**
  - P5.16.1 Generalità.
  - P5.16.2 Specifiche funzionali di alto livello.
- P5.17. Interfaccia utente ipertestuale.**
  - P5.17.1 Generalità.
  - P5.17.2 Specifiche funzionali di alto livello.
- P5.18. Modulo "ModelSynth".**

**P6. Riferimenti bibliografici.**

## **Premessa generale.**

Il presente documento costituisce un aggiornamento dei precedenti documenti tecnici della Stazione di Lavoro Musicale Intelligente, concernenti l'architettura e le specifiche funzionali, illustrate rispettivamente nei Rapporti Tecnici N7/09 e N7/53.

Parte del documento è stata tratta dai citati rapporti, specialmente per quanto riguarda la descrizione di moduli non modificati in seguito.

Il presente rapporto fornisce una descrizione dettagliata dell'architettura complessiva della SLMI (Stazione di Lavoro Musicale Intelligente) e dei singoli moduli software di cui è costituita.

## **P0. Introduzione.**

Nei successivi paragrafi viene fornita una descrizione ad alto livello dell'architettura complessiva della SLMI e dei singoli moduli software di cui è costituita.

In particolare, sono descritte:

- \* l'architettura software complessiva della SLMI;
- \* le funzionalità di alto livello che possono configurarsi come ambiti operativi specifici per l'utente finale, sia in relazione ai moduli propri della SLMI che in relazione a moduli applicativi reperibili in commercio;
- \* le specifiche funzionali dei singoli moduli software della SLMI;
- \* le librerie di modelli, dati e programmi sviluppate e di corredo per la SLMI;
- \* gli standard di codifica e di ambiente di sviluppo software adottati per il progetto SLMI.

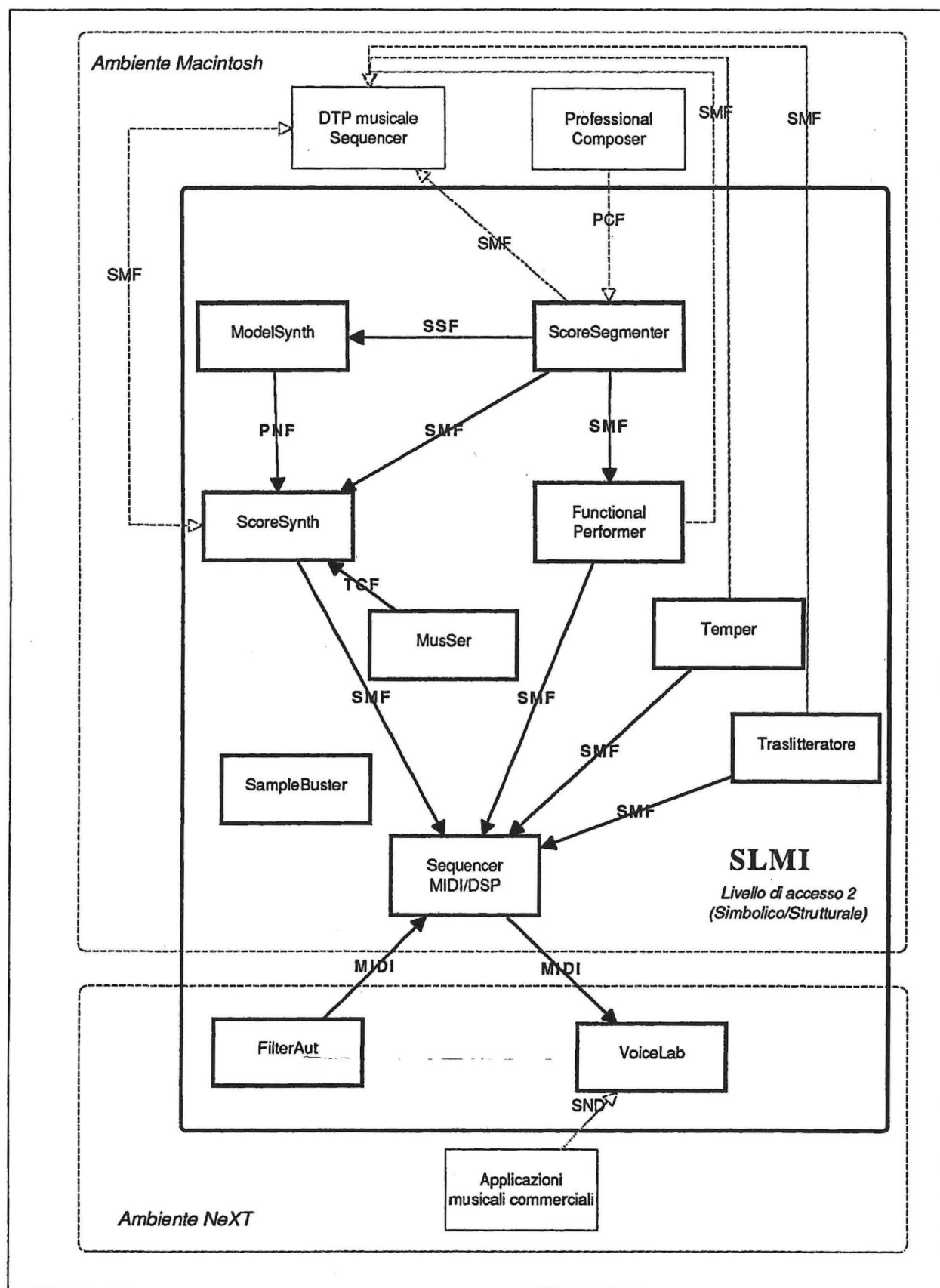
## **P1. Architettura della SLMI.**

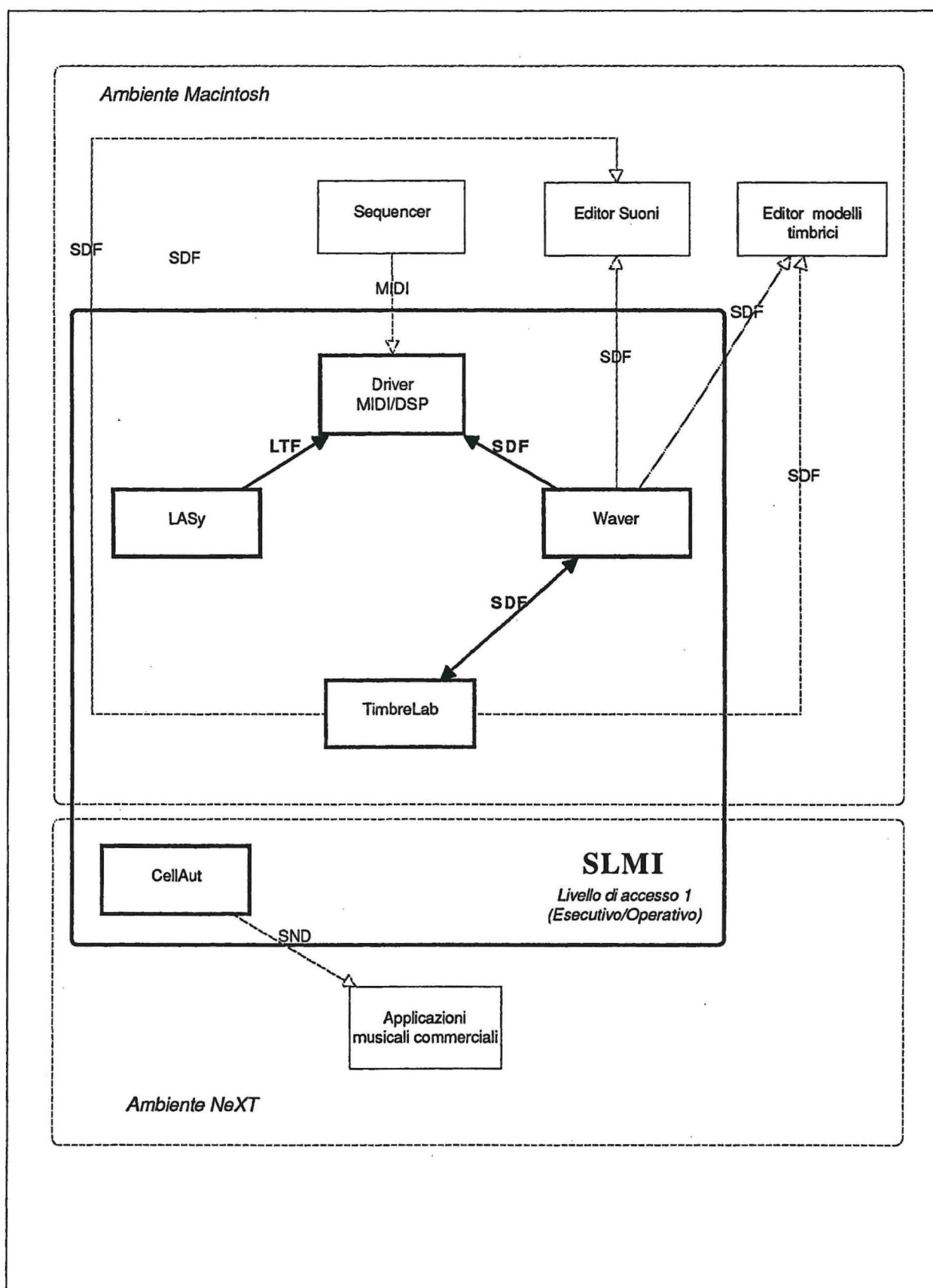
La SLMI si configura come un insieme di moduli hardware e software organizzati in modo gerarchico.

In prima approssimazione, il sistema è strutturato a quattro livelli:

0. **Livello hardware:**
  - Sistema di elaborazione general purpose.
  - Moduli per la elaborazione numerica di suoni.
  - Dispositivi per la registrazione e la riproduzione del suono.
  - Dispositivi di controllo e distribuzione di segnali (patchbay MIDI e di segnali audio, mixer).
1. **Livello S/W 1:**
  - Moduli a basso livello per le attività di analisi-sintesi del suono, la costruzione e l'elaborazione di campioni e modelli timbrici, la produzione di master audio numerici e il restauro di registrazioni audio.
2. **Livello S/W 2:**
  - Moduli ad alto livello per le attività di composizione, analisi/sintesi di testi musicali e di strutture di testi musicali, performance musicale e audio visuale, orchestrazione, DTP musicale.
3. **Interfaccia utente:**
  - Interfaccia utente ad alto livello di tipo ipertestuale.

Nel diagramma funzionale della pagina seguente sono rappresentati i livelli S/W 1 e 2.





Nel diagramma precedente sono riportati i collegamenti funzionali esistenti all'interno della SLMI. A parte i comandi MIDI, che attualmente costituiscono l'unico tipo di comunicazione fra i diversi ambienti hardware/software in cui è stata sviluppata la SLMI (Macintosh e NeXT), i moduli comunicano mediante file il cui tipo di formato è specificato nel seguente elenco:

- \* SMI: Standard MIDI File;
- \* PNF: file prodotti e letti da ScoreSynth contenenti modelli di Reti di Petri;
- \* TCF: file di formato Tab Converter;
- \* PCF: file di formato Professional Camposer;
- \* SSF: file prodotti da ScoreSegmenter risultanti da un'operazione d'analisi;
- \* SDF: file di formato Sound Designer;
- \* LTF: file prodotti da LASy Workbench contenenti i parametri di sintesi LASy;
- \* SND: file sonori di formato NeXT.

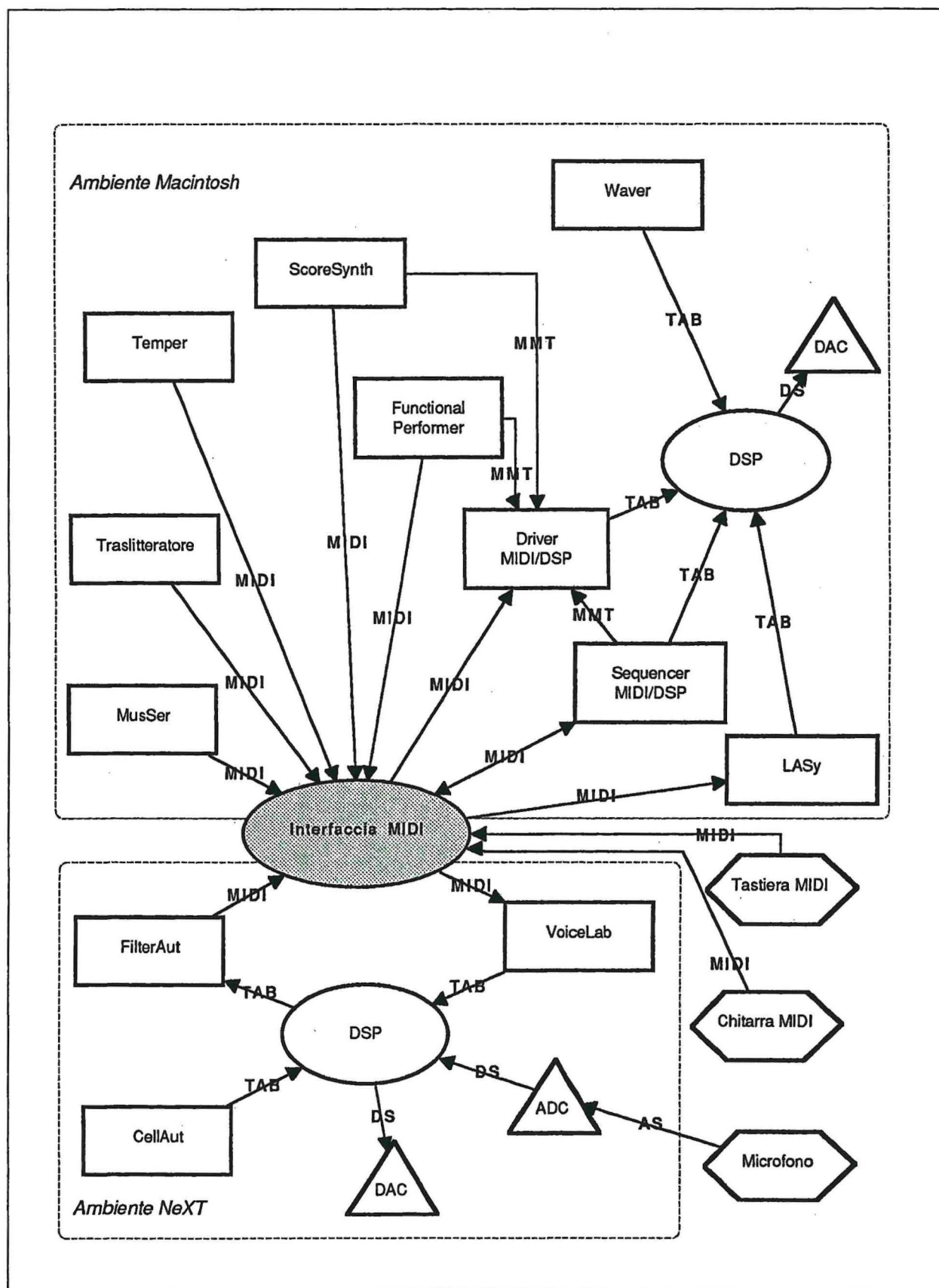
**Avvertenze:** - I file di comunicazione fra il modulo WAVER e il modulo Driver MIDI/DSP, pur essendo di formato Sound Designer, non contengono segnali digitali bensì i parametri di sintesi ottenuti mediante WAVER.

Come già accennato in precedenza, oltre ai file, i moduli della SLMI comunicano fra loro e con l'esterno mediante canali di informazione alternativi (sia logici che fisici), quali ad esempio i messaggi MIDI.

Tale tipo di comunicazione coinvolge principalmente i moduli caratterizzati da un'esecuzione in tempo reale, durante la quale possono essere coinvolti dispositivi quali tastiere e chitarre MIDI, microfoni, mixer, amplificatori e casse acustiche.

Il diagramma riportato in seguito illustra, ad alto livello, le funzionalità operanti in tempo reale della SLMI, i messaggi presenti nei canali di comunicazione sono:

- \* MIDI: messaggi MIDI;
- \* TAB: parametri di sintesi;
- \* DS/AS: segnale digitale/analogico;
- \* MMT: collegamento virtuale fra interfacce MIDI effettuato mediante il MIDI Management Tool.





I collegamenti illustrati sono in verità virtuali in quanto all'interno della SLMI alcuni moduli entrano in concorrenza specialmente per quanto riguarda l'utilizzo del DSP, tale vincolo porta alle seguenti conclusioni:

- \* Waver, LAsy e il Driver MIDI/DSP richiedono l'uso del medesimo DSP, non è possibile quindi eseguirli contemporaneamente;
- \* ScoreSynth, MusSer, Traslitteratore, Temper, Functional Performer non possono essere utilizzate contemporaneamente sul medesimo calcolatore in quanto richiedono una continua interazione con l'utente e quindi un continuo utilizzo del display;
- \* Il Sequencer MIDI/DSP può comunicare mediante il MIDI Management Tool (MMT) con tutte le applicazioni che possiedono tale tipo di interfaccia, ad esempio i moduli della SLMI : ScoreSynth, Functional Performer e il Driver MIDI/DSP. Quest'ultimo può essere utilizzato parallelamente in quanto il Sequencer MIDI/DSP opzionalmente disattiva il possesso della scheda;
- \* Il Driver MIDI/DSP comunica mediante MMT con tutte le applicazioni interfacciabili, ad esempio all'interno della SLMI con i moduli ScoreSynth, Functional Performer e Sequencer MIDI/DSP;
- \* FilterAut, CellAut, VoiceLab richiedono contemporaneamente l'uso dell'unico DSP disponibile attualmente in ambiente NeXT, ciò esclude l'esecuzione parallela sul medesimo calcolatore;
- \* L'interfaccia MIDI mette in comunicazione le applicazioni agenti su diversi calcolatori, è possibile infatti un'eventuale comunicazione fra i due ambienti controllando ad esempio VoiceLab mediante il Sequencer MIDI/DSP.

Il livello "Interfaccia utente" è costituito da un insieme di moduli intercambiabili che opereranno su un insieme di stack attivi di HyperCard, per le seguenti attività: gestione delle richieste utente; presentazione di risultati/diagnostiche in formati standard; presentazione di oggetti/entità ad un livello di astrazione adatto ad un utente che intende operare in un particolare sotto ambiente.

## **P2. Discussione schematica delle funzionalità della SLMI.**

In questo paragrafo sono brevemente discusse le funzionalità di alto livello della SLMI che possono configurarsi come ambiti operativi specifici per l'utente finale, sia in relazione ai moduli propri della SLMI che in relazione a moduli applicativi reperibili in commercio.



### *Composizione*

L'attività di composizione musicale può anche essere realizzata combinando la scrittura tradizionale resa possibile dai programmi *composer* commerciali insieme con la scrittura non tradizionale basata su reti di Petri, oggetti musicali e algoritmi con il modulo *ScoreSynth*. Per la composizione di opere musicali di natura seriale è stato realizzato *MusSer*, un modulo specifico per la generazione di serie musicali equilibrate con un qualsivoglia temperamento equabile a  $n$  gradi (con  $n$  pari). Gli oggetti musicali argomento degli algoritmi definiti nei modelli *ScoreSynth* possono essere derivati tanto da moduli applicativi commerciali via Standard MIDI File che dal modulo *MusSer* o anche dal modulo *ScoreSegmenter* che decompone in oggetti musicali intere partiture precedentemente codificate.

E' in corso la realizzazione del modulo *ModelSynth*, il quale, ricevendo in ingresso il risultato della computazione di *ScoreSegmenter*, sintetizza reti di Petri.

### *Analisi/sintesi testi musicali*

La trasformazione di testi musicali preesistenti è una peculiare funzionalità della SLMi resa possibile dall'uso combinato dei moduli *ScoreSegmenter* (per l'analisi della partitura e la sua decomposizione in oggetti musicali), *ScoreSynth* (per la sintesi di trasformazioni della partitura mediante modelli di reti di Petri) e *Functional Performer* (per la sintesi di trasformazioni della partitura mediante operatori in tempo reale).

Il modulo *ModelSynth* costituisce il collegamento responsabile della comunicazione fra i moduli *ScoreSynth* e *ScoreSegmenter*.

### *Analisi strutture testi musicali*

Il modulo *ScoreSegmenter*, come già accennato qui sopra, consente di identificare gli oggetti musicali di cui è composta una partitura con una particolare affidabilità per i brani musicali in forma di fuga o di sonata. E' quindi uno strumento essenziale e fortemente innovativo di supporto al musicologo e in generale allo studioso dei testi musicali.

### *Performance musicale*

Il modulo *Functional Performer* permette di eseguire brani musicali attivando da tastiera ASCII funzioni musicali che vengono applicate ad

oggetti musicali predisposti precedentemente o tramite moduli SLMI o tramite composer o sequencer commerciali in formato Standard MIDI File.

#### *Performance audiovisuale*

Il modulo *TEMPER* permette di eseguire automaticamente processi audio visuali costituiti da animazioni grafiche nello stile del pittore M. C. Escher e processi MIDI ottenuti per traslitterazione delle animazioni grafiche (interpretate come piani melodici).

#### *Analisi-sintesi suono*

In ambiente Macintosh è disponibile un modulo (microprogramma per DSP Motorola 56001) per l'analisi e la sintesi con la trasformata di Wavelet. In combinazione con programmi come Digidesign SoftSynth e Digidesign SoundDesigner è possibile confrontare e potenziare le caratteristiche della analisi-sintesi additiva per armoniche con quelle della analisi-sintesi mediante trasformata wavelet.

In ambiente NeXT è situato il modulo di VoiceLab dedicato alla sintesi del parlato. Esso è in grado di effettuare operazioni di analisi sul contenuto di Sound File formato NeXT provenienti da altre applicazioni, è disponibile inoltre il modulo *FilterAut*, il quale effettua la stima della frequenza della fondamentale di un segnale analogico.

#### *Preparazione modelli timbrici*

E' disponibile un ambiente sperimentale, in ambiente Macintosh, formato da un insieme di moduli (alcuni dei quali utilizzano dei microprogrammi per il DSP Motorola 56001) per la sintesi del suono secondo diversi modelli di sintesi.

*Waver* (Ambiente sperimentale per microprogrammi DSP per Sound Accelerator Card) consente di sperimentare in tempo reale tre tecniche di sintesi: Funzioni in due variabili, Waveshaping e l'uso della trasformata di Wavelet.

*LASy Workbench* implementa la sintesi sonora LASy (tecnica di sintesi che utilizza automi cellulari) in tempo reale.

Il modulo *Driver MIDI/DSP* realizza diverse tecniche di sintesi quali Waveshaping, Funzioni in due variabili, LASy, ricevendo in ingresso comandi MIDI in tempo reale attraverso il MMT.

*TimbreLab* consiste in un sistema ad architettura aperta per la programmazione timbrica che permette la realizzazione di tutte le tecniche

di sintesi presenti in letteratura, differisce dai precedenti non permettendo l'ascolto del segnale prodotto in tempo reale.

In ambiente NeXT è disponibile il modulo *CellAut* dedicato alla sintesi sonora LASy in tempo reale.

### *Orchestrazione*

L'attività di orchestrazione è una delle più complesse per il musicista; richiede un ciclo ipotesi-di-orchestrazione/sperimentazione-sonora/editing-dell'assetto-orchestrale che si ripete tante volte quante sono necessarie per il raggiungimento dell'obiettivo di tessitura timbrica ricercata; per questo è stato progettato un ambiente in cui siano operativi concorrentemente due tipi di moduli software: uno di supporto alla ricerca dei suoni nella banca di timbri disponibili (il modulo *SampleBuster* o gli ambienti per la sperimentazione timbrica nella SLMI) e l'altro per l'esecuzione dei brani nelle varie ipotesi di tessitura timbrica (un qualunque sequencer MIDI commerciale con il Driver MIDI/DSP o il modulo *Sequencer MIDI-DSP* nella SLMI, che consentono di controllare contemporaneamente unità di sintesi MIDI e DSP microprogrammati in tempo reale).

### *DTP musicale*

Integrando i moduli della SLMI capaci di generare partiture (*ScoreSynth*, *Functional Performer*) con gli strumenti per il Desk Top Publishing musicale commerciali (ad es. il programma Finale della Coda Music Software), è possibile ottenere le partiture generate in notazione tradizionale e ai massimi livelli di qualità (fotocompositrici che accettano codici PostScript; es. Linotronics 100 P).

### *Produzione master audio numerici*

I programmi sequencer commerciali, il modulo *Sequencer MIDI-DSP* e alcuni programmi e dispositivi di utilità (Digidesign DAT I/O, Digidesign SoundTools) consentono di registrare, prima a livello MIDI e poi come audio digitale su disco rigido SCSI, su nastro DAT e come audio analogico su nastri open reel, il master delle produzioni musicali.

### P3. Librerie disponibili nella SLMI.

I moduli della SLMI hanno accesso in lettura e scrittura a un insieme di librerie che costituiscono uno degli elementi costitutivi essenziali della SLMI stessa; in questo paragrafo vengono succintamente descritte tali librerie; nel Rapporto tecnico R3 ("Specifiche funzionali dei moduli della STAZIONE DI LAVORO MUSICALE INTELLIGENTE") saranno dettagliati i vincoli relativi agli accessi per ogni singolo modulo.

#### *Libreria di reti di Petri "macro"*

Il modulo *ScoreSynth* ha accesso ad una libreria di reti di Petri "macro" ovvero un insieme di costrutti di reti di Petri che sono descritti in forma parametrica e che possono essere invocati mediante parametri formali per valore. Questa libreria è stata progettata per rendere disponibile quell'insieme di costrutti che ricorrono sovente nello sviluppo di modelli di partiture con reti di Petri e rendere perciò più efficiente la fase di editing dei modelli formali.

#### *Libreria di routine di lettura/scrittura di Standard MIDI File v 1.0*

Questa libreria (costituita da una cinquantina di routine) costituisce un vero e proprio modulo della SLMI e come tale sarà descritta in R3 e più succintamente nel prossimo paragrafo. E' stata progettata per rendere disponibile in forma standard a tutti i ricercatori del presente progetto il principale strumento di comunicazione con i moduli applicativi reperibili in commercio e tra i moduli della SLMI stessi. Tanto le applicazioni commerciali quanto i moduli della SLMI hanno funzionalità di import/export di file a livello di partitura MIDI come Standard MIDI File v 1.0.

#### *Libreria di patch timbriche per microprogrammi del DSP Motorola 56001*

Il modulo *Sequencer MIDI-DSP* consente di variare parametricamente il modello timbrico corrente mediante patch timbriche; ogni microprogramma sviluppato per tale modulo ha a disposizione un corredo di patch timbriche (analogamente a quanto accade per i preset dei sintetizzatori MIDI) organizzati in una libreria di 128 suoni accessibile dal sequencer. L'utente di quest'ultimo potrà ovviamente sviluppare le sue specifiche patch timbriche e aggiungerle alla libreria sviluppata nell'ambito del presente progetto.

### *Primitive Sonore*

Il modulo di TimbreLab utilizza un insieme di funzioni mediante le quali è possibile realizzare tutte le tecniche di sintesi sonora.

La libreria è strutturata in cinque sezioni le quali evidenziano i diversi utilizzi durante la fase di programmazione timbrica:

- 1) Generatori di onde inviluppo
- 2) Generatori di rumore
- 3) Oscillatori
- 3) Tool
- 5) Funzioni di nuova creazione.

Le prime tre sezioni hanno un significato intuitivo, la sezione dedicata ai tool comprende tutte le funzioni agenti sul segnale prodotto dai generatori: sincronizzatori, interruttori, funzioni di ritardo, filtri...

L'ultima sezione contiene attualmente qualche esempio, essa è dedicata alle funzioni realizzate dall'utente.

### *Libreria di patch timbriche del Driver MIDI/DSP*

Il modulo *Driver MIDI-DSP* ha introdotto un ampliamento del formato Patch timbriche per microprogrammi (Patch File). Ora è possibile avere contemporaneamente a disposizione 128 serie (bank) di 128 suoni ciascuno, sempre organizzati in una libreria accessibile al *Driver MIDI/DSP*. E' da sottolineare che mentre il *Driver MIDI/DSP* legge i Patch File del Sequencer MIDI/DSP essendone un ampliamento, quest'ultimo non può leggere i Patch File del Driver a meno che questo non sia a bank singolo. Tramite il *Driver MIDI/DSP* l'utente potrà ovviamente sviluppare nuove patch e aggiungerle alla libreria sviluppata nell'ambito del presente progetto.

### *Banca timbrica di campioni sonori*

Il modulo *SampleBuster* è basato su una banca timbrica di campioni sonori costituita da circa 7.000 elementi delle seguenti famiglie timbriche originali:

- f1) MUMS (McGill University Master Samples)
- f2) Sound Examples (Current Directions in Computer Music Research, M.I.T.)
- f3) AKAI S900 Samples Library
- f4) AKAI S950 Samples Library
- f5) AKAI S1000 Samples Library
- f6) libreria di suoni campionati del L.I.M.

Le famiglie timbriche f1 ed f2 sono memorizzate su CD-DA (Compact Disc Digital Audio); le famiglie f3, f4, f5 ed f6 sono memorizzate su dischi magnetici. Tali campioni non sono disponibili all'interno della stazione di lavoro in quanto non costituiscono materiale originale del Laboratorio di Informatica Musicale, ma provengono dall'esterno.

Ovviamente è possibile utilizzare e organizzare campioni provenienti da qualsiasi biblioteca, essendo i criteri ed i formati scelti per la caratterizzazione dei campioni di carattere assolutamente generale.

#### **P4. Standard utilizzati nell'implementazione.**

Al fine di rendere i moduli della SLMi integrati pienamente con i moduli applicativi reperibili in commercio è stato convenuto di adottare un insieme di standard per la codifica dei dati e per lo sviluppo del software di cui i seguenti sono i più rilevanti:

- s1) *MIDI Management Tool* : come ambiente standard per lo sviluppo di moduli che utilizzano l'interfaccia MIDI sotto Multifinder (ambiente multitasking per Macintosh); consente di connettere logicamente in tempo reale i flussi MIDI in ingresso e in uscita ai/dai vari moduli della SLMi e dei moduli commerciali implementati nel medesimo ambiente; in tal modo i vari moduli si scambiano flussi di dati in tempo reale consentendo così di costruire applicazioni più complesse combinando opportunamente i singoli moduli;
- s2) *Standard MIDI File v.1.0* : come standard per la codifica dei file contenenti partiture a livello MIDI; consente di scambiare tali informazioni tra tutti i moduli della SLMi e con i moduli commerciali che operano a livello di partitura MIDI;
- s3) *Digidesign Soundfile* : come standard per la codifica di dati di tipo sonoro (campioni sonori, master digitali); consente di scambiare dati di tipo sonoro tra tutti i moduli di editing, di registrazione, di sintesi e di analisi del suono in forma numerica.

#### **P5. Specifiche funzionali dei moduli della SLMi.**

Nel presente paragrafo, per ciascun modulo sperimentale appartenente alla SLMi sviluppato dall'unità operativa dell'Università degli Studi di Milano, vengono riportate le descrizioni dettagliate,



unitamente alle specifiche funzionali di alto livello definite mediante i criteri esposti in (Camurri et al., 1989, parte II, §P8).

**Avvertenze:** - 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.

## **P5.1. Modulo "ScoreSynth 3.0".**

### **P5.1.1 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.

L'utente ha anche a disposizione un ambiente che permette l'esecuzione passo a passo di una rete. All'interno di questo ambiente di debugging è possibile, durante l'esecuzione di un modello, verificare le distribuzioni delle marche e le trasformazioni degli oggetti.

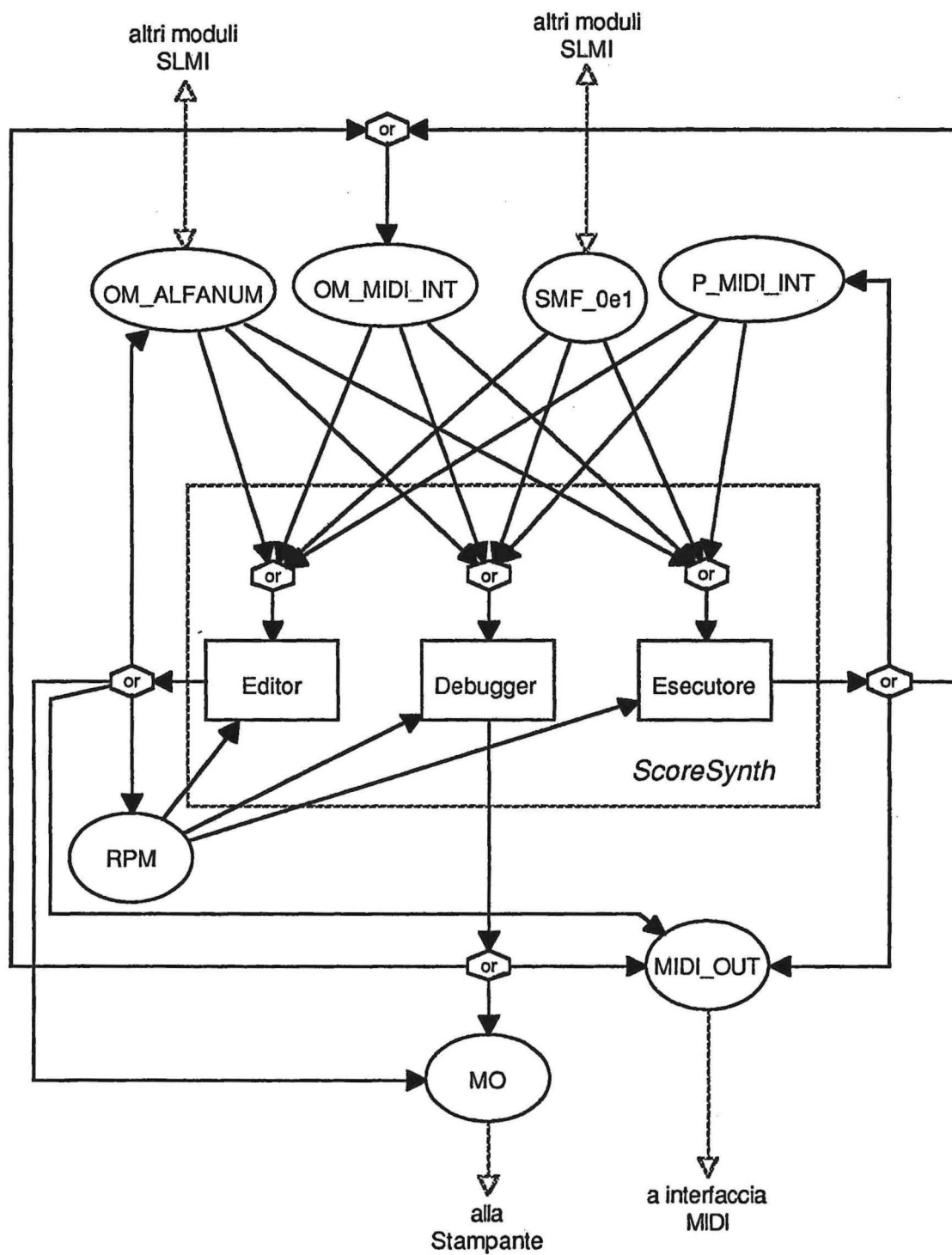
Per quanto riguarda la comunicazione con l'"esterno" si è resa possibile l'importazione e l'esportazione di file in formato Standard MIDI File (SMF) 0/1, ormai universalmente diffusi, così da garantire un'interfaccia verso tutti i programmi commerciali di sequencing e di stampa di partiture musicali. 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).



## P5.1.2 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 MUSIC (M/37, 1992).

## P5.2. Modulo "MusSer".

### P5.2.1 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.

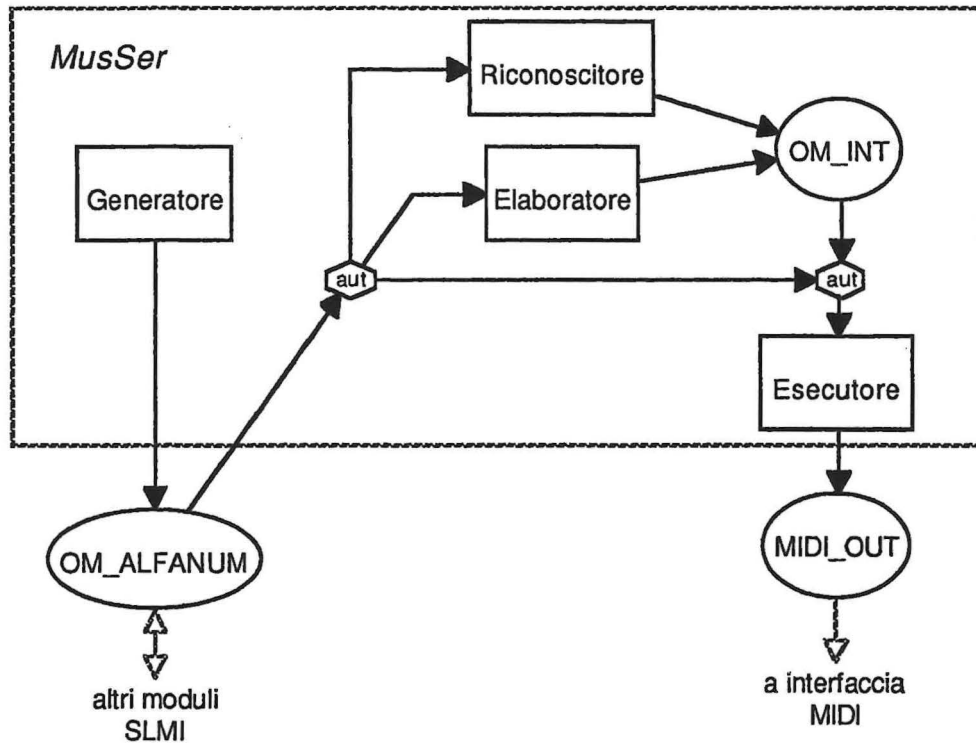
*MusSer* legge e scrive file in formato TabConverter (i dati musicali sono rappresentati in forma di sequenze alfanumeriche contenenti tutte le informazioni presenti usualmente nelle partiture). Questi file possono rappresentare archivi di serie dodecafoniche generati e/o elaborati da *MusSer*, oppure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

## P5.2.2 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 MUSIC (M08, 1991).

### **P5.3. Modulo "Traslitteratore".**

#### **P5.3.1 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).

### P5.3.2 Specifiche funzionali di alto livello.

#### Diagramma funzionale.

vedi pagina 24

#### Descrizione funzionale in pseudocodice ADA-like.

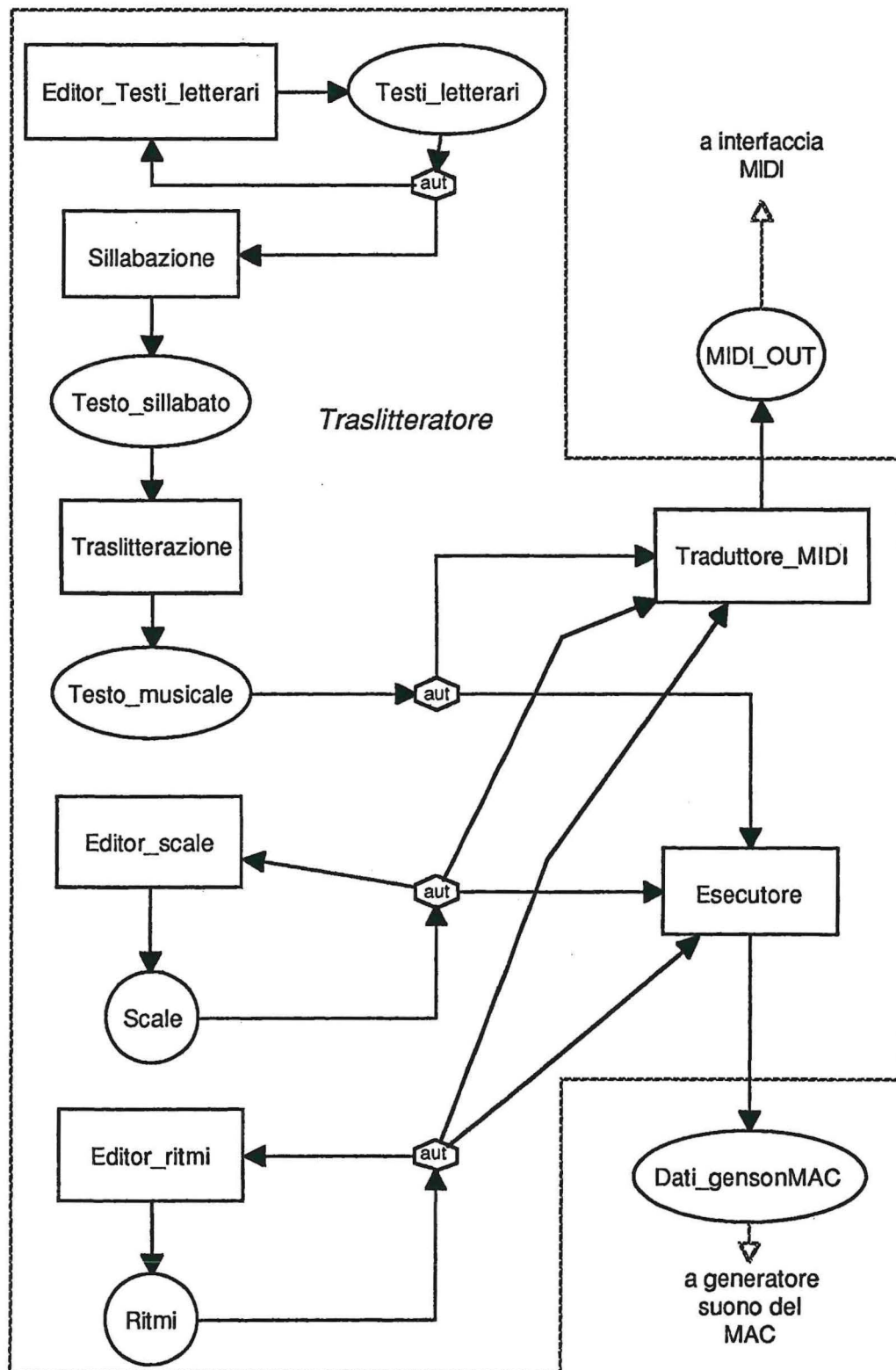
```
-- Descrizione modulo Traslitteratore

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 Dati_gensonMAC;
--> Dati per il generatore di suoni del Macintosh
type Testo_sillabato;
--> Testo letterario suddiviso in sillabe e pronto per la
--> fase di traslitterazione
type Testo_musicale;
--> Dati musicali ottenuti dalla traslitterazione
type MIDI_OUT;
--> Dati standard MIDI diretti a interfaccia MIDI
```





Per quanto riguarda i dettagli implementativi e le modalità d'uso del modulo *Traslitteratore* si faccia riferimento al corrispondente documento operativo MUSIC (M20, 1991).

#### P5.4. Modulo "Functional Performer".

##### P5.4.1 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).

Le strutture musicali costituenti le primitive su cui operare sono definite come *frammenti melodici* cioè sequenze di note musicali caratterizzate da altezza, durata e intensità.

Ai frammenti melodici sono associati gli *oggetti grafici* che ne permettono la manipolazione da parte dell'utente attraverso gli operatori di trasformazione implementati.

Gli *operatori di trasformazione* modificano i frammenti selezionati fornendo il risultato della loro applicazione in tempo reale.

Il modulo accetta in input una serie di frammenti musicali codificati in forma di Standard MIDI File e, sfruttando le routine ed il ciclo di lettura di SMF del modulo *Libreria di routine 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.

Esiste la possibilità di accedere ad archivi personali o a librerie di terzi, prelevare i frammenti melodici necessari per la performance ed iniziare l'esecuzione selezionando frammenti e operatori attraverso i comandi presenti sullo schermo. I comandi sono impartiti attraverso i dispositivi di input previsti dall'unità centrale, tipicamente una tastiera ed un mouse.

I risultati delle trasformazioni sono trasmessi serialmente ad una interfaccia MIDI alla quale sono collegate una o più periferiche in grado di interpretare i dati elaborati, tipicamente processori di segnali MIDI.

Durante l'esecuzione è possibile selezionare un frammento melodico presente in memoria, ed inserirlo nella partitura all'istante di tempo desiderato. Il frammento viene suonato nelle sue caratteristiche originali se non viene selezionato alcun operatore, nel caso venga selezionata una qualsiasi trasformazione tra quelle implementate il frammento verrà suonato con le nuove caratteristiche.

Gli operatori a disposizione attuano sui frammenti melodici le seguenti trasformazioni: inversione speculare di altezze o di dinamiche, dilatazione o contrazione di intervalli o di durate, rotazione di altezze o dinamiche o durate, trasposizione in altezza verso il registro acuto o verso il registro grave, accelerazione o rallentamento dell'esecuzione, modelli per la trasformazione di dinamiche, retrogradazione di altezze o di durate o di dinamiche o totale, fusione di frammenti.

Le modalità di inserimento dei frammenti, o delle loro trasformazioni, sono essenzialmente due: nella prima il frammento viene accodato all'ultimo frammento da eseguire (FIFO), mentre nella seconda l'utente dispone l'oggetto associato al frammento su una griglia temporale, in questo caso il frammento verrà eseguito all'istante corrispondente.

Al termine della performance, i dati relativi a un'esecuzione possono essere codificati in SMF, in questo modo i dati relativi alla partitura generata non vengono solamente inviati all'interfaccia MIDI, ma vengono conservati in memoria per la scrittura su disco in formato SMF.

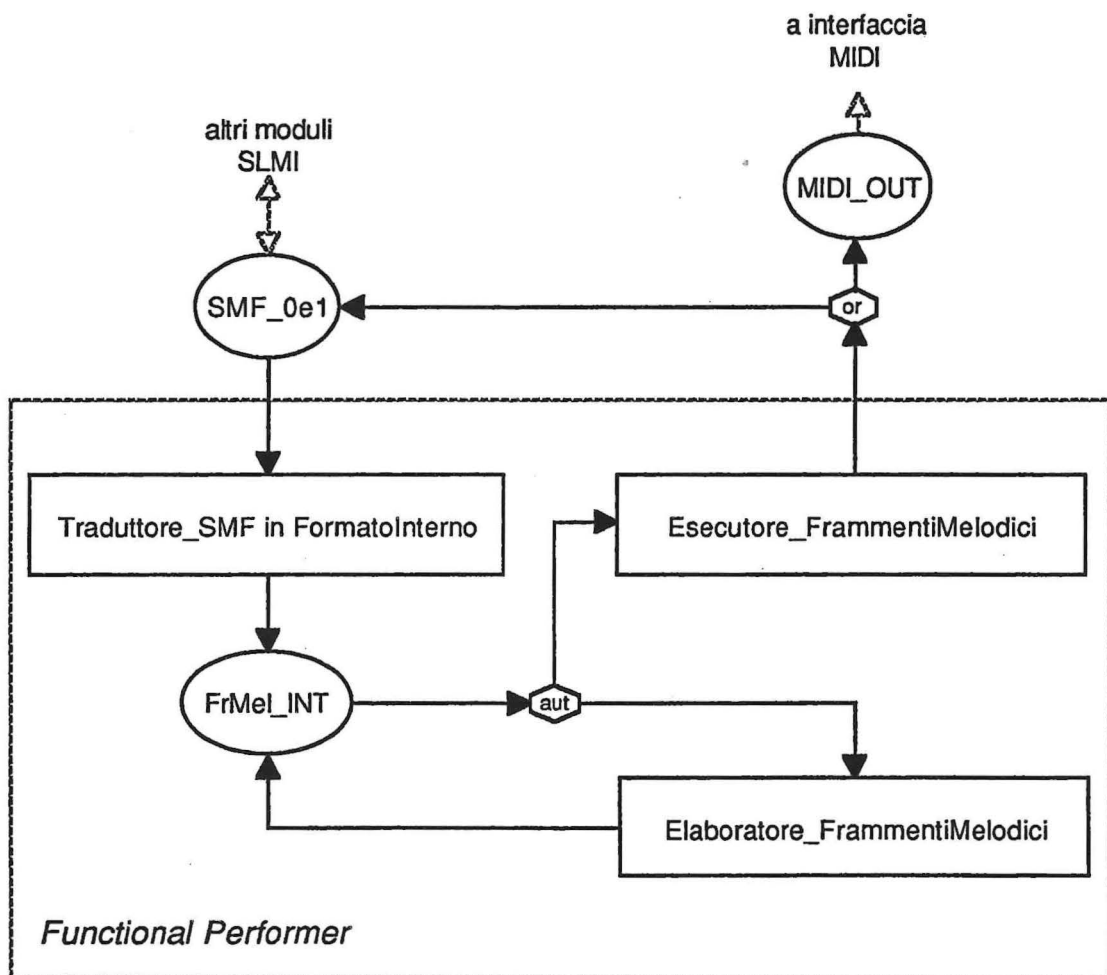
Nella prima fase della realizzazione del modulo sono state delineate le caratteristiche e la funzionalità della nuova applicazione, individuando gli operatori di trasformazione di maggior interesse musicale; nel contempo è stata impostata la rappresentazione grafica dell'informazione, con particolare riferimento alla ergonomia della disposizione dei comandi.

alla luce del fatto che i tempi di interazione di una performance possono essere estremamente serrati.

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).

#### P5.4.2 Specifiche funzionali di alto livello.

##### Diagramma funzionale.



### 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 MUSIC (M40, 1992).

## P5.5. Modulo "TEMPER".

### P5.5.1 Generalità.

Il modulo *TEMPER* è stato studiato per tradurre in musica immagini grafiche di un tipo particolare, denominato tassellazioni (Morini, 1987) (Haus & Morini, 1992); 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.

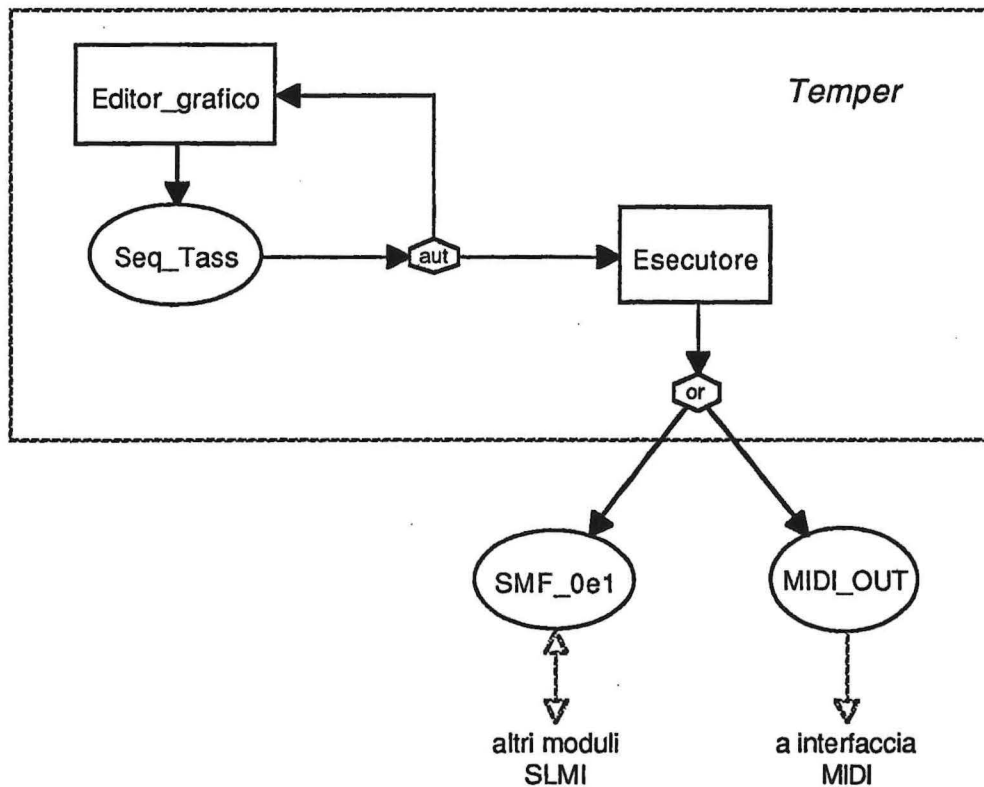
La generazione del suono è realizzata mediante dispositivi MIDI. *TEMPER*, inoltre, può anche dirigere il suo output musicale su Standard MIDI File.

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).



## P5.5.2 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 MUSIC (M33, 1992).

## P5.6. Modulo "ScoreSegmenter".

### P5.6.1 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 modo 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.

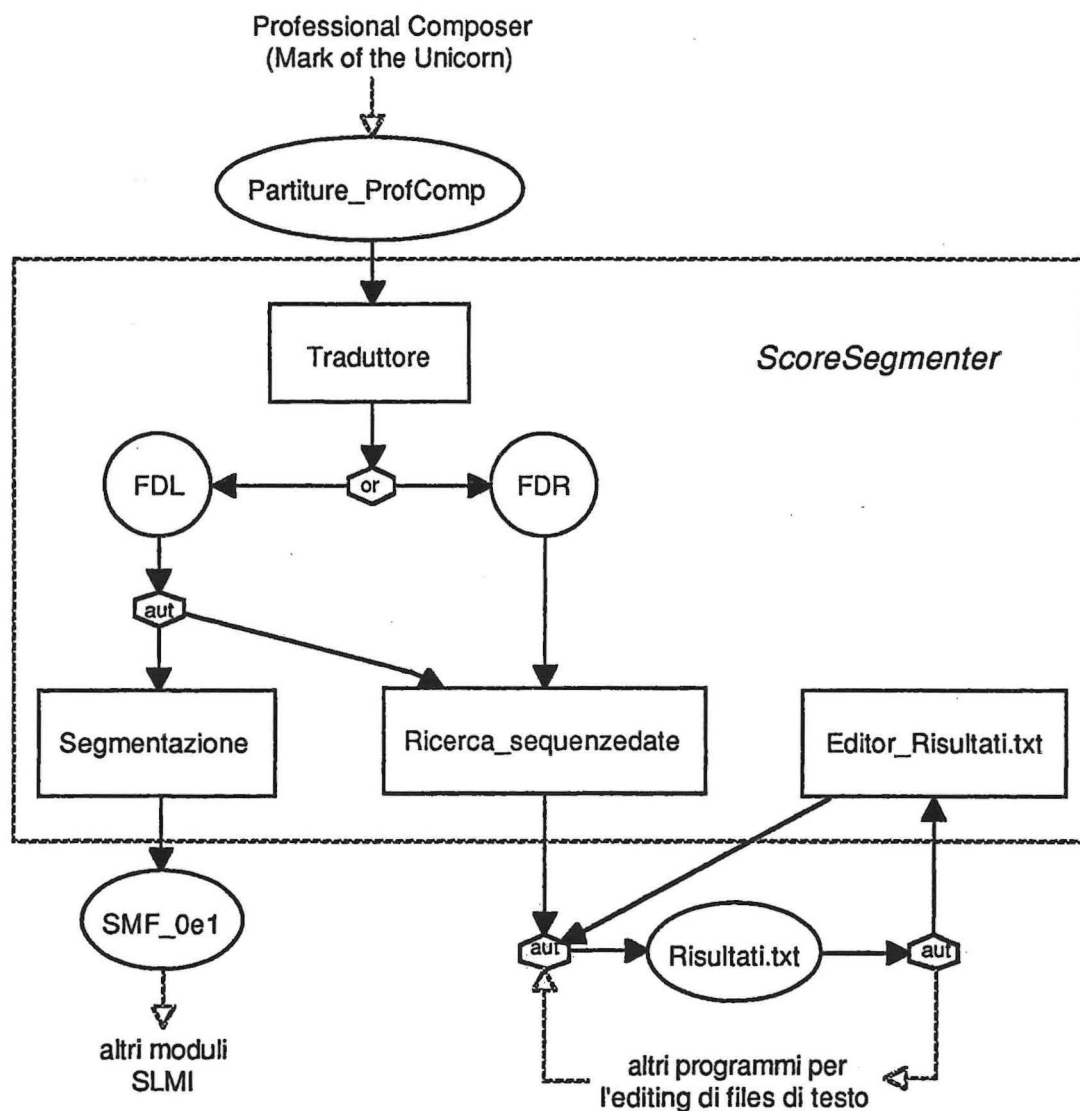
Per quanto riguarda la segmentazione, ci si è basati sulla forma musicale della fuga, generalizzandola poi per le altre forme. Per la individuazione degli oggetti ci si è basati non solo sulle loro ripetizioni, che sono evidentemente un aspetto necessario ma non sufficiente ai fini musicali. Un tema 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).

#### P5.6.2 Specifiche funzionali di alto livello.

##### Diagramma funzionale.



### 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_ProfComp;
--> Partiture codificate nel formato Professional 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 MUSIC (M18, 1991).

#### P5.7. Modulo "SampleBuster".

##### P5.7.1 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 *SampleBuster* 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).

## P5.7.2 Specifiche funzionali di alto livello.

### Diagramma funzionale.

vedi pagina 37.

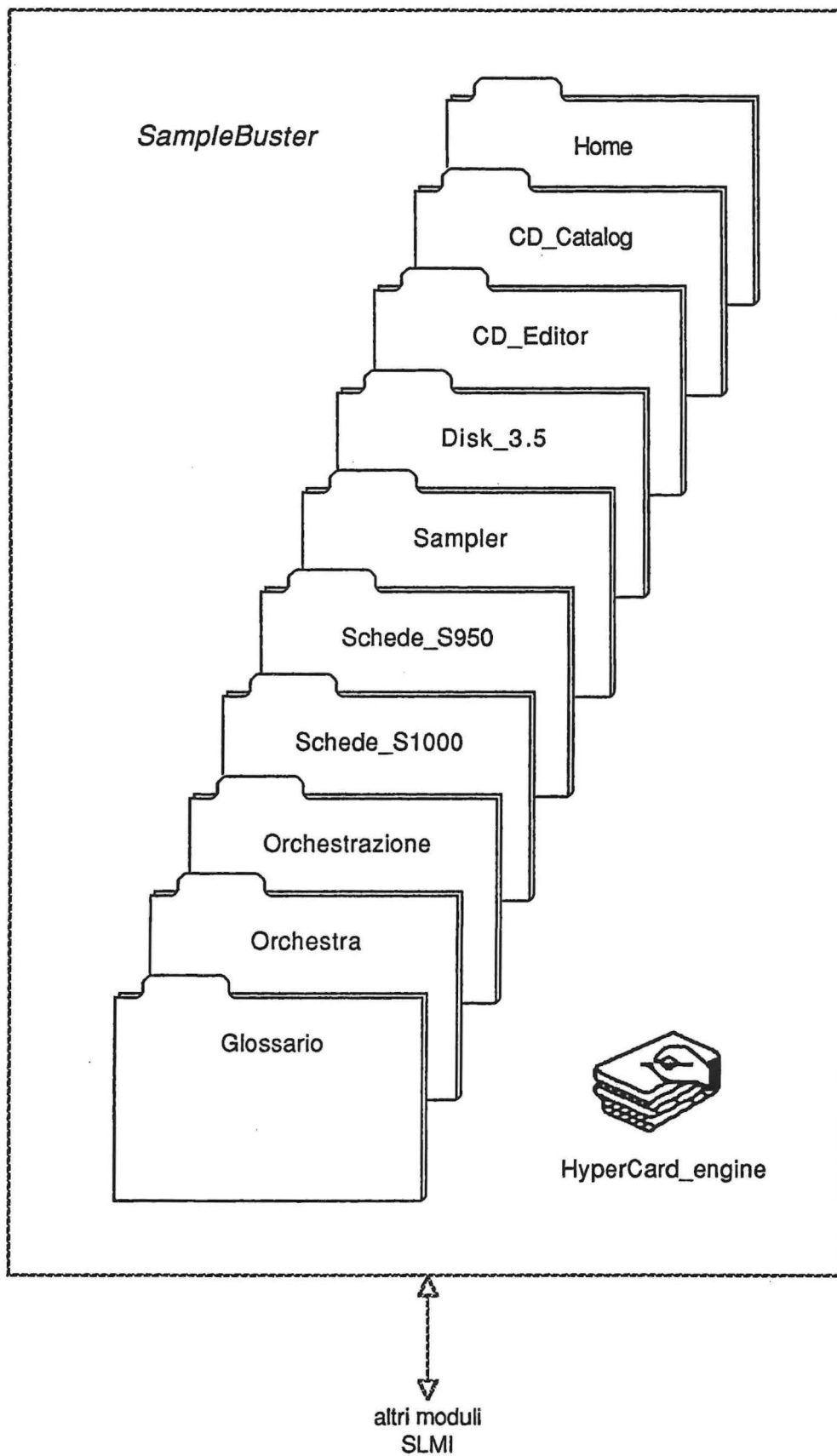
### Descrizione funzionale in pseudocodice ADA-like.

```
-- 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 MUSIC (M13, 1991).

## P5.8. Modulo "Sequencer MIDI/DSP".

### P5.8.1 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).

#### P5.8.2      Specifiche funzionali di alto livello.

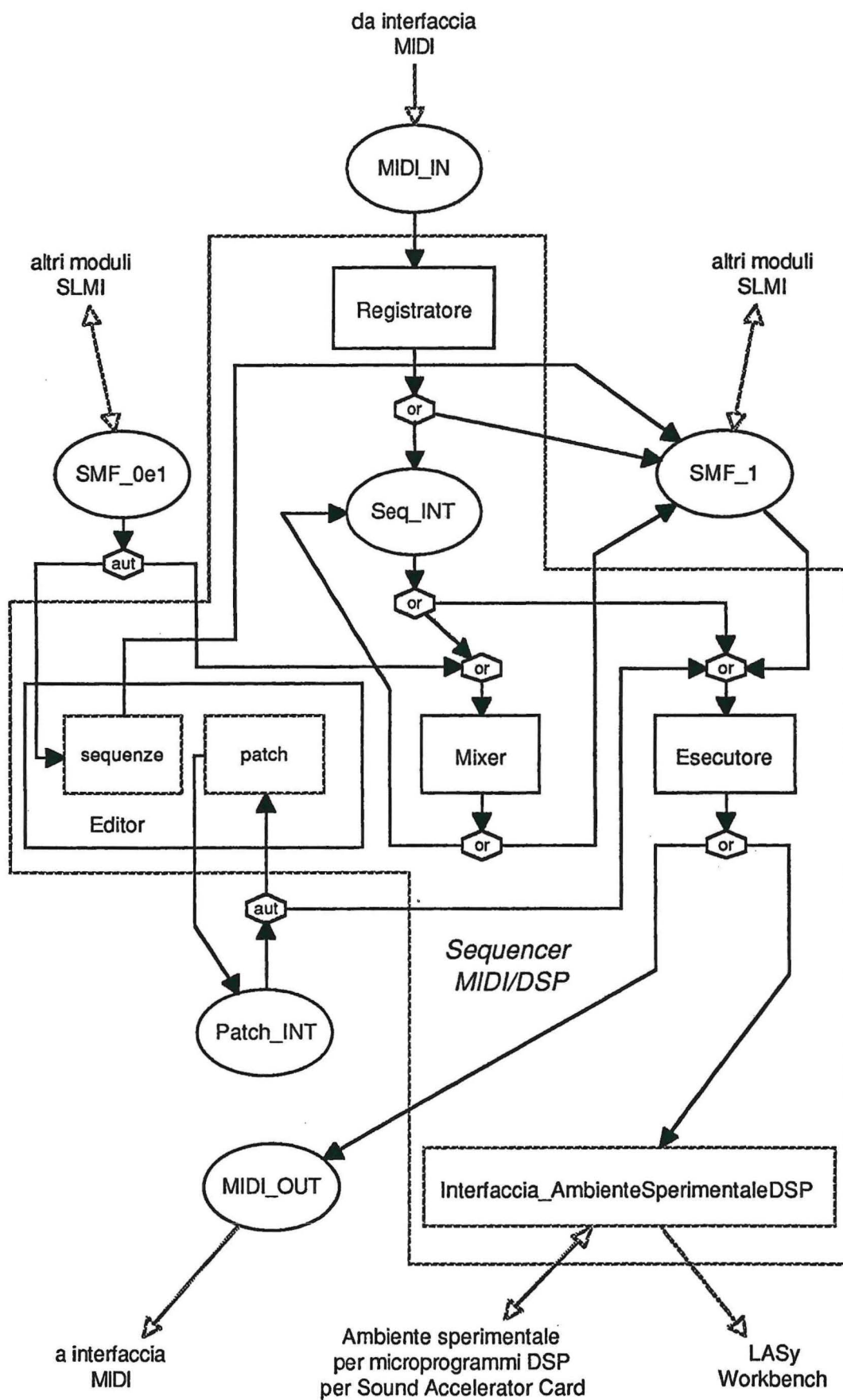
##### Diagramma funzionale.

vedi pagina 40.

##### 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 MUSIC (M05, 1991).

## P5.9. Modulo "Microprogrammi per Sequencer MIDI/DSP e Sound Accelerator Card".

### P5.9.1 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) (Mitsuhashi, 1982) (Borgonovo & Haus, 1986), sintesi mediante Waveshaping (WSH) (Roads,

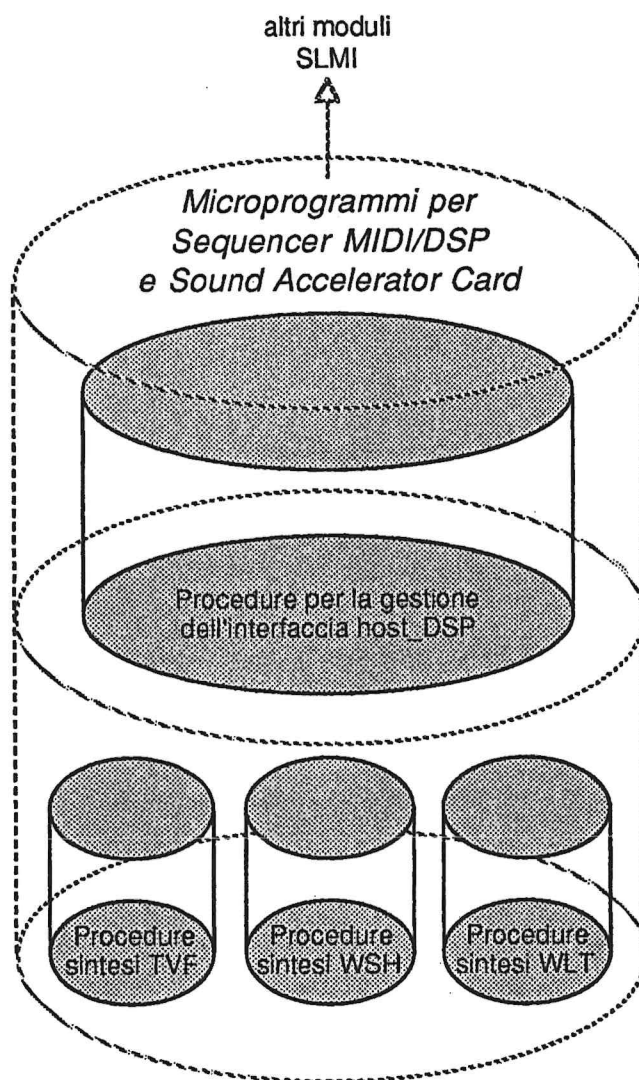


1979) e sintesi mediante trasformata Wavelet (WLT) (Dutilleux, Grossmann & Kronland-Martinet, 1988).

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.

#### P5.9.2 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 MUSIC (M10, 1991).

#### **P5.10. Modulo "Libreria di routine di lettura/scrittura di Standard MIDI File 1.0".**

##### **P5.10.1 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.

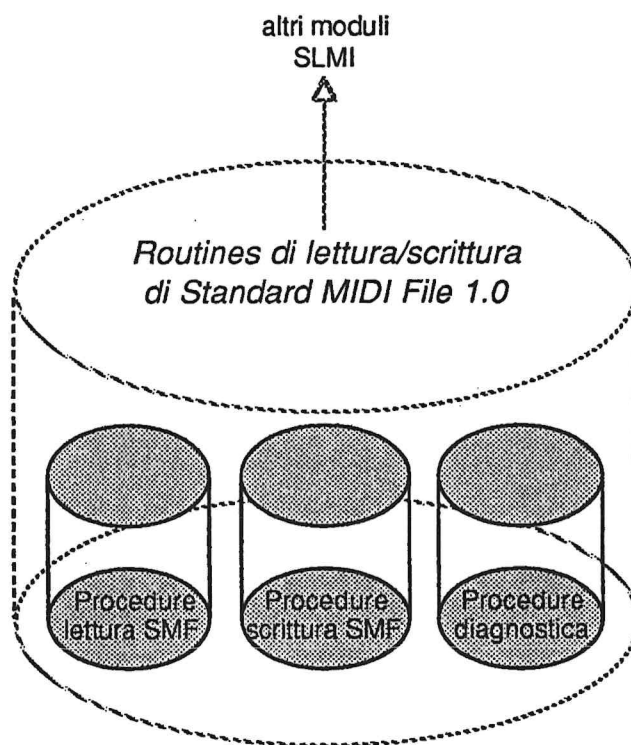
Il modulo *Libreria di routine di lettura/scrittura di Standard MIDI File 1.0* (Bettitoni, 1990) è costituito da routine scritte in PASCAL, utilizzabili in ambiente MPW (Macintosh Programmer's Workshop), per la scrittura/codifica e per la lettura/decodifica di Standard MIDI File (versione

1.0) nei formati 0 e 1, secondo le specifiche fornite dalla *International MIDI Association* (IMA, 1988). In particolare comprende routine per la codifica e la decodifica di numeri espressi in forma di "variable-length quantity" (rappresentazione "compressa" di valori numerici utilizzata negli SMF), routine per la lettura e la scrittura di header generali e header di traccia, routine di lettura/decodifica e codifica/scrittura di metaeventi, routine di lettura/decodifica e codifica/scrittura di eventi, routine di supporto per la diagnostica.

Nell'ambito della SLMi, il modulo *Libreria di routine 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).

#### P5.10.2 Specifiche funzionali di alto livello.

##### Diagramma funzionale.



### 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 *Routine di lettura/scrittura di Standard MIDI File 1.0* si faccia riferimento al corrispondente documento operativo MUSIC (M04, 1991).

#### **P5.11. Modulo "Ambiente sperimentale per microprogrammi DSP per Sound Accelerator Card".**

##### **P5.11.1 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).

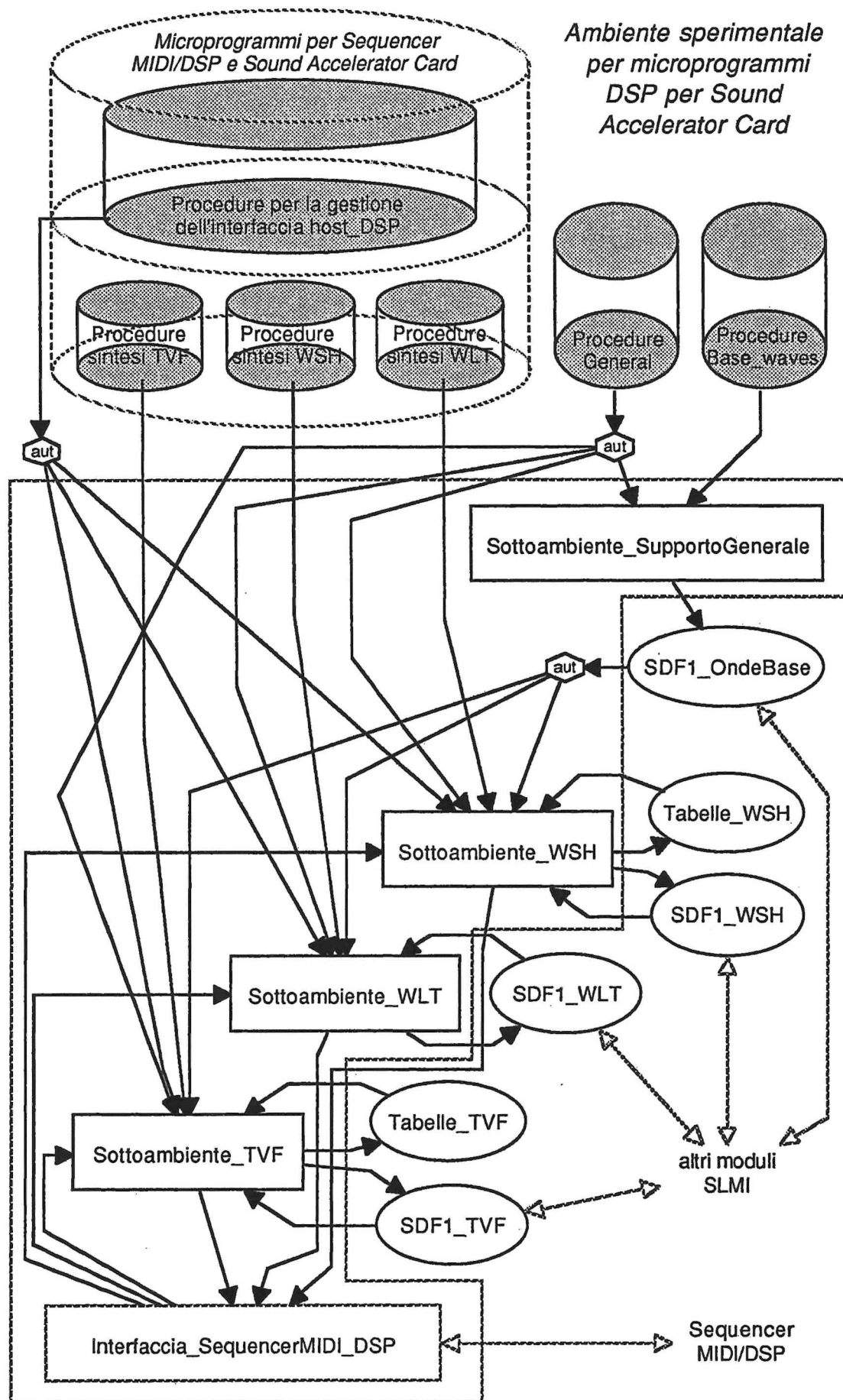
## P5.11.2 Specifiche funzionali di alto livello.

### Diagramma funzionale.

vedi pagina 44.

### 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. SP4.2  
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 MUSIC (M15, 1991).

## P5.12. Modulo "LASy Workbench".

### P5.12.1 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 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 controller.

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).

```

graph TD
    TFO(Tabelle_FdO) --> CET[Creazione_Editing_Tabelle]
    TFO --> CS[Costruzione_Strumenti]
    TRD(Tabelle_RdT) --> CET
    TRD --> CS
    PS(Parametri_Strumenti) --> CS
    CET --> CS
    CS --> DS(Dati_Strumenti)
    DS --> AUT1{aut}
    PS --> AUT1
    AUT1 --> S[Sintesi]
    S --> D_DSP(Dati_DSP)
    D_DSP -.-> DSP[a DSP scheda SA]
    D_DSP --> AUT2{aut}
    DS --> AUT2
    AUT2 --> CRT(Controlli_RT)
    CRT --> GCM[Gestione_Controlli_MAC]
    CRT --> GCMIDI[Gestione_Controlli_MIDI]
    GCM --> CRT
    GCMIDI --> CRT
    CRT --> S
    S --> ISMD[Interfaccia_SequencerMIDI_DSP]
    ISMD --> SM[Sequencer MIDI/DSP]
    ISMD --> AUT2
    CM(Controlli_MAC) -.-> GCM
    CM -->|da controlli Mac (tastiera alfanumerica, mouse)| CM
    MI(MIDI_IN) -.-> GCMIDI
    MI -->|da interfaccia MIDI| MI
  
```

### 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 MUSIC (M30, 1992).

## P5.13. Modulo "TimbreLab".

### P5.13.1 Generalità.

Il modulo di TimbreLab consiste in un sistema ad architettura aperta, sviluppato in ambiente LabVIEW su computer Macintosh, dedicato alla produzione di segnali digitali.

A differenza di LASy e Waver, TimbreLab permette la realizzazione di tutte le tecniche di sintesi disponibili in letteratura, il tempo computazionale relativo alla produzione di un segnale digitale é però proporzionale alla lunghezza espressa in numero di campioni del segnale stesso (tale vincolo deriva dalla assoluta generalità richiesta durante la fase di progetto).

TimbreLab é stato progettato in base alla seguente schematizzazione: il modello timbrico può essere inteso come la composizione di moduli classificabili in quattro classi:

- a) *moduli controllori*: responsabili del controllo e della sincronizzazione dei moduli presenti nel modello;
- b) *moduli generatori*: i quali generano segnali con forma d'onda fissata;
- c) *moduli operandi*: agenti sulle forme d'onda più semplici, essi possono essere ad uno o più ingressi provenienti dai generatori;
- d) *moduli consumatori*: la cui configurazione dei parametri in entrata stabilisce il formato e la destinazione finale del segnale prodotto.

Ogni modulo dialoga con il mondo esterno esclusivamente mediante le porte di ingresso e di uscita, attraverso le quali passano due tipi di informazione:

- a) le funzioni di controllo dei moduli;
- b) il segnale generato.

Diversificando i flussi di informazione fra i moduli è possibile realizzare agevolmente tutte le tecniche di sintesi sonora note in letteratura; l'oggetto risultante da una sessione di progettazione sperimentale di sintesi è chiamato *strumento*.

Ogni strumento a sua volta può far parte come modulo generatore di un modello timbrico più complesso (modelli ricorsivi non sono ammessi), ogni classe precedentemente illustrata infatti può essere arricchita con nuovi elementi.

La necessità di rivolgersi ad utenti inesperti in ambiti informatici (quali generalmente i musicisti) ha fatto sì che TimbreLab fosse progettato affinché l'utente, mediante una penna ideale, potesse codificare lo strumento progettandone mediante strumenti grafici (icone e patch) la forma d'onda.

Affinché questo fosse possibile è stato scelto come ambiente di sviluppo LabVIEW 2.2, il quale utilizza un linguaggio grafico (G Language) formato da moduli a carattere funzionale rappresentati da icone. La fase di sviluppo ha coinvolto altri linguaggi oltre a quello grafico, quali il C e l'Assembler per l'interfaccia con la scheda sonora.

Sono stati realizzati altri moduli oltre quelli di base, al fine di rendere la programmazione di uno strumento il più agevole possibile e favorire contemporaneamente la formazione di nuovi moduli da inserire nella libreria.

Ogni strumento, una volta in esecuzione, produce un segnale digitale. Questo può essere *ascoltato*, *visualizzato* nel dominio del tempo e delle frequenze e *salvato* su disco in un formato compatibile con altre applicazioni di carattere musicale (Sound Designer, TurboSynth...).

Codificato con le opportune modalità, lo strumento diventa una funzione di libreria che l'utente costruisce secondo le proprie esigenze.

Sono perciò tre i tipi di oggetti software generati con il sistema sviluppato:

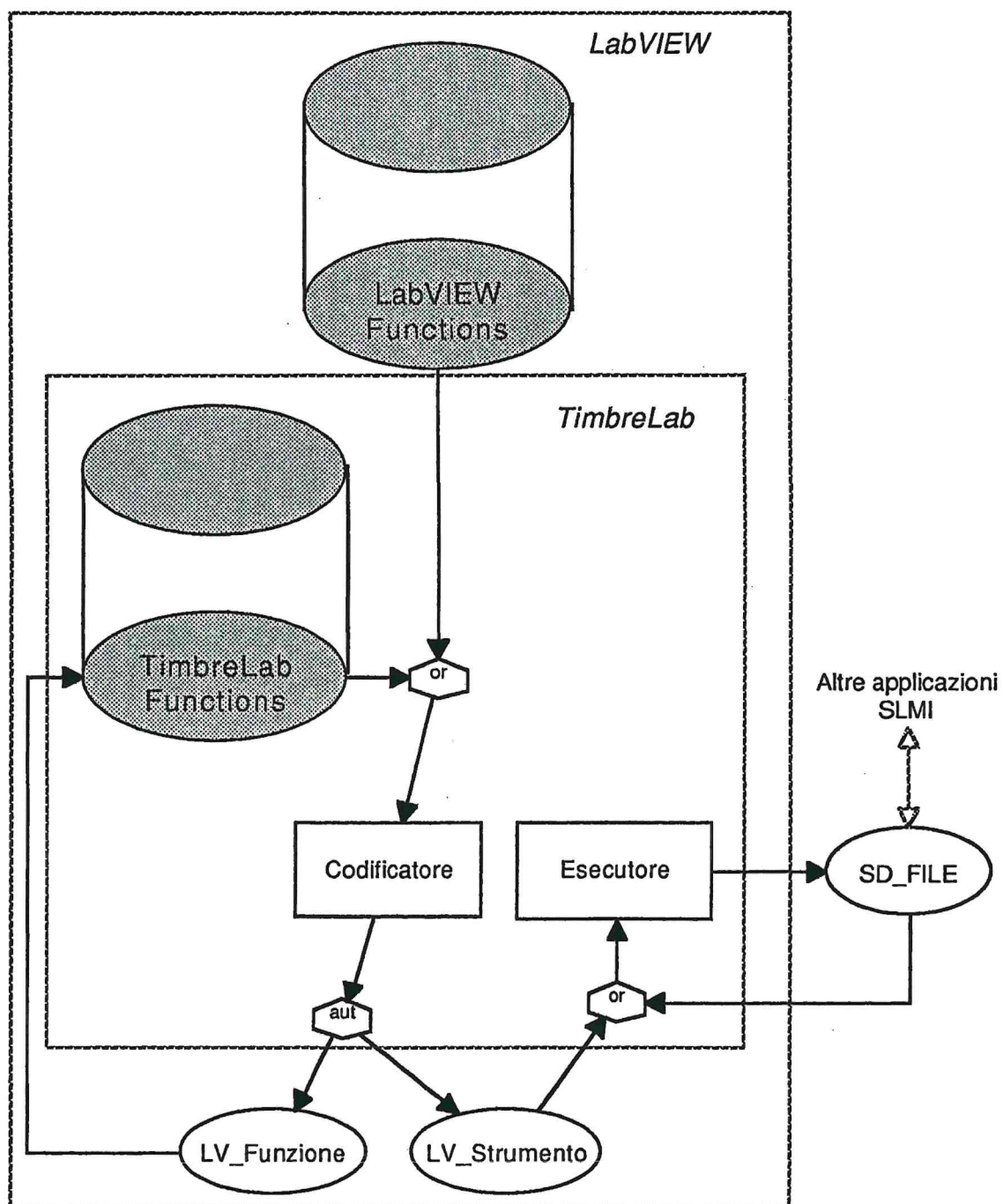
- a) *lo strumento*;
- b) *il file sonoro*;
- c) *modulo di libreria*;

Potenzialmente TimbreLab risulta un efficace mezzo di analisi in quanto ogni strumento durante l'esecuzione fornisce contemporaneamente la schematizzazione grafica del segnale generato (il codice), il segnale digitalizzato stesso (l'output), il relativo spettro di potenza e la rappresentazione nel dominio del tempo, entrambi con possibilità di zoom.

Nell'ambito della SLMI, il modulo *TimbreLab* 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 *Preparazione di modelli timbrici* (Camurri et al., 1989, parte I, §P2).



Diagramma funzionale.





## Descrizione funzionale in pseudocodice ADA-like.

- - Descrizione modulo *TimbreLab*

```
package TimbreLab is
    procedure Codificatore
    procedure Esecutore
end TimbreLab
```

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

```
type LV_Funzione;
- -> File contenente funzioni codificate in GLanguage
type LV_Strumento;
- -> File contenente strumenti codificati in GLanguage
type SD_File;
--> Sound File Standard DigiDesign
```

Per quanto riguarda i dettagli implementativi e le modalità d'uso del modulo *TimbreLab* si faccia riferimento al corrispondente documento operativo MUSIC (M28, 1991).

### P5.14. Modulo "VoiceLab".

#### P5.14.1 Generalità.

*VoiceLab* è il risultato di un lavoro di tesi che costituisce un'indagine nel campo della sintesi della voce in generale e del parlato in particolare, con controllo dell'andamento intonativo del parlato sintetico. Questa applicazione costituisce quindi una base per sintetizzare anche il canto, che altro non è se non un caso particolare di intonazione. *VoiceLab* è eseguibile in ambiente NeXT ed è costituita da una parte di codice che viene eseguita sul processore principale e da un modulo di sintesi del segnale vocale scritto in microcodice che gira su DSP Motorola 56001, di serie su tutti i sistemi NeXT.

VoiceLab permette di creare e modificare un insieme di voci, intendendo con "voce" un gruppo di caratteristiche che descrivono un parlatore sintetico, e di sintetizzare il parlato da testo scritto.

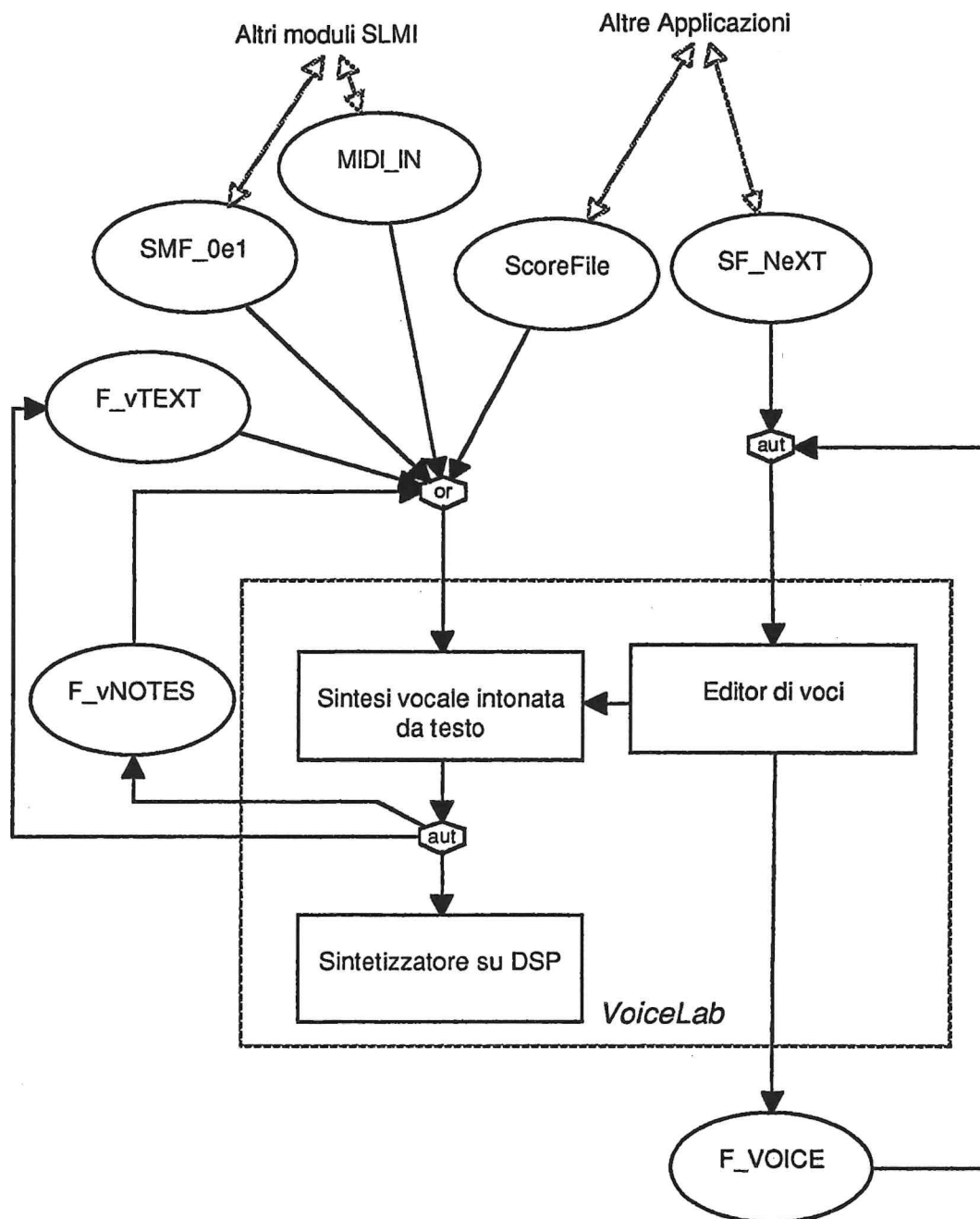
Più precisamente, VoiceLab permette di:

- \* definire l'insieme dei fonemi costituenti una voce, il loro tipo e la loro durata;
- \* estrarre parametri LPC (Linear Predictive Coding, Markel e Gray 1976) da una voce campionata, per poterli utilizzare in una successiva fase di sintesi;
- \* definire per i fonemi un timbro completamente sintetico, scegliendo la posizione e la larghezza di banda dei picchi formantici (Rabiner e Schafer 1978) nel dominio delle frequenze, ascoltando in tempo reale l'effetto dei cambiamenti e vedendo una FFT del segnale generato in una opportuna finestra;
- \* poter scegliere l'enfasi da porre sugli accenti, la percentuale di allungamento dei suoni a causa del raddoppiamento sintattico, la durata dei silenzi, la velocità ed il volume del parlato ed altri parametri che descrivono il comportamento di un parlatore;
- \* sintetizzare il parlato da testo scritto in tempo reale, con la possibilità di agire sul testo da sintetizzare a vari livelli, da quello del gruppo di parole alla parola singola alle unità di sintesi che la costituiscono, per cambiare le durate e creare e modificare gli andamenti di intonazione ed intensità del suono a vari livelli di dettaglio;
- \* definire andamenti di intonazione a partire da dati standard MIDI acquisiti da una periferica MIDI esterna, da Standard MIDI file o da Scorefile, con la possibilità di associare gruppi di note a gruppi di vocali, e sintetizzare il canto in tempo reale in base a dati standard MIDI provenienti da una periferica MIDI esterna.

Nell'ambito della SLMI, il modulo *VoiceLab* 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 *Preparazione di modelli timbrici* (Camurri et al., 1989, parte I, §P2).

P5.14.2 Specifiche funzionali di alto livello.

Diagramma funzionale.



### Descrizione funzionale in pseudocodice ADA-like.

- - Descrizione modulo *VoiceLab*

**package** *VoiceLab* **is**

**procedure** Editor di voci

**procedure** Sintesi vocale intonata da testo

**procedure** Sintetizzatore su DSP

**end** *VoiceLab*

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

**type** MIDI\_IN;

- -> Dati standard MIDI provenienti da interfaccia MIDI

**type** SMF\_0e1;

- -> Standard MIDI File formato 0 e 1

**type** ScoreFile;

--> Standard di descrizione testuale di una partitura

**type** SF\_NeXT;

- -> Sound File Standard di tipo NeXT

**type** F\_vTEXT;

- -> File di formato interno decrivente un testo

**type** F\_vNOTES;

--> File di formato interno contenente una sequenza di note

**type** F\_VOICE;

--> File di formato interno descrivente una voce

Per quanto riguarda i dettagli implementativi e le modalità d'uso del modulo *VoiceLab* si faccia riferimento al corrispondente documento operativo MUSIC (M35, 1992).

### **P5.15. Modulo "Cellular Automata Sound Processor".**

#### **P5.15.1 Generalità.**

Il modello matematico degli automi cellulari è stato introdotto da von Neumann e da Ulam negli anni '60 come idealizzazione di sistemi fisici (von Neumann, 1963), e successivamente applicati ad una vasta serie di ambiti, dai sistemi biologici ai problemi della teoria dei numeri, oltre a

essere studiati come modelli matematici di per sé, ad esempio (Wolfram, 1986).

Tutti questi studi evidenziano le caratteristiche di complessità dei risultati ottenuti, caratteristiche che li rendono interessanti per l'applicazione alla sintesi timbrica. Un algoritmo che utilizza gli automi per la questa sintesi è Linear Automata Synthesis, o LASy, sviluppata da Jacques Chareyron presso il Laboratorio di Informatica Musicale, e che è stato implementato nelle due applicazioni *CellAut* e *FilterAut*.

LASy opera a partire da uno stato iniziale le cui celle vengono inizializzate con i valori di una semplice forma d'onda, ad esempio un periodo della funzione seno, e produce una successione di stati che rappresenta la descrizione del segnale digitale prodotto, corrispondente all'evoluzione dell'onda di innesco operata dall'automa.

CellAut permette di:

- \* sintetizzare un segnale utilizzando l'algoritmo LASy a partire da una forma d'onda e una funzione di transizione scelte dall'utente;
- \* modificare graficamente la funzione di transizione;
- \* utilizzare una seconda funzione di transizione da mescolare alla prima;
- \* utilizzare come forma d'onda iniziale un frammento di un suono campionato;
- \* modificare i vari parametri durante la sintesi, ascoltando immediatamente i risultati;
- \* salvare il risultato della sintesi come Standard NeXT Sound File, e collegarsi ad altre applicazioni per la sua analisi.

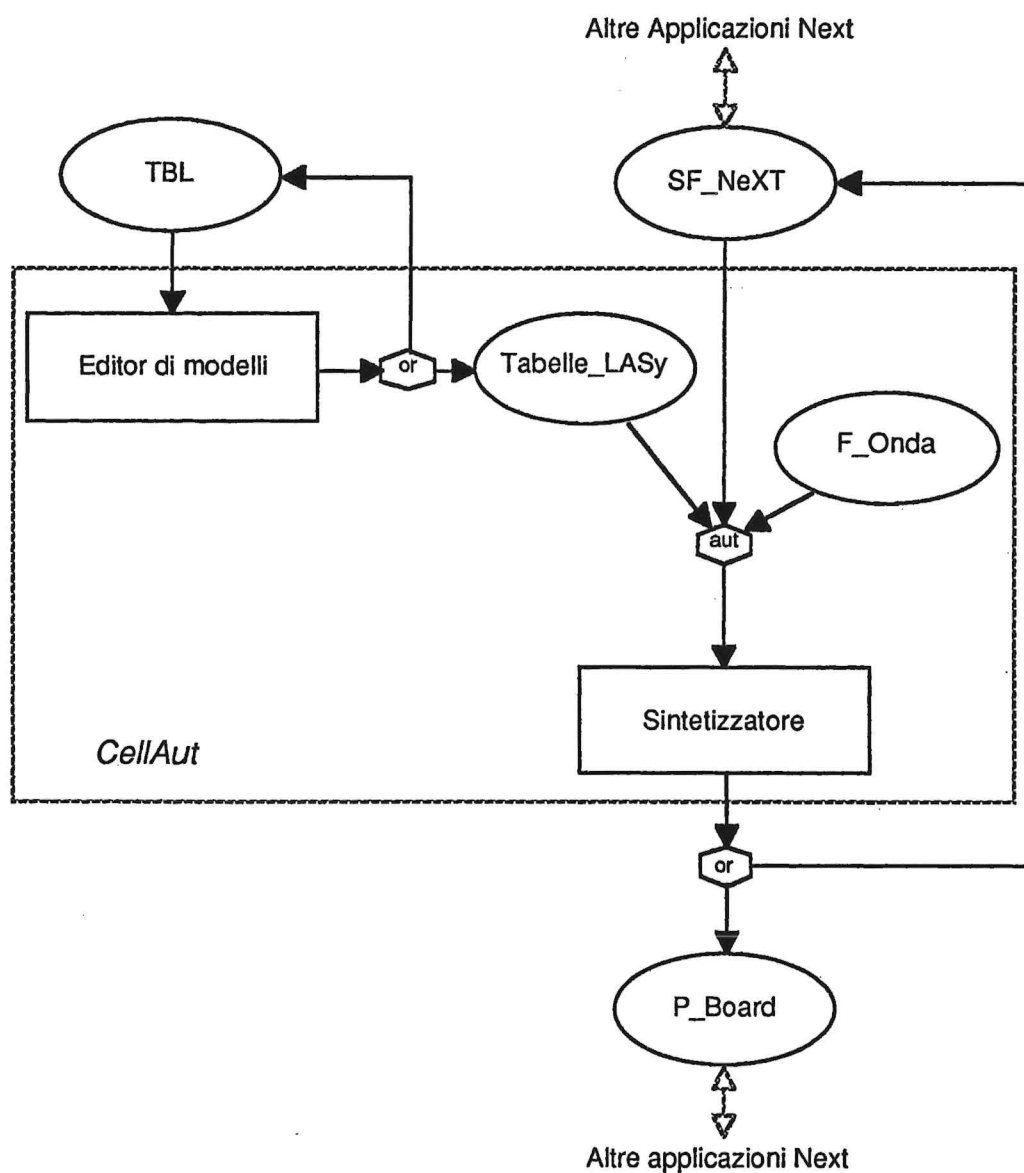
L'utente ha un controllo duplice sull'evoluzione della sintesi: i parametri che controllano la funzione di transizione sono modificabili tramite oggetti grafici presenti sullo schermo del calcolatore, e l'altezza del suono prodotto è controllata dall'altezza di un segnale analogico rilevato da un microfono, ad esempio il tono di voce dello stesso utente.

Parallelamente a CellAut, è stata sviluppata una seconda applicazione, chiamata FilterAut, che consente di filtrare in tempo reale un segnale analogico e contemporaneamente di variare i parametri di filtraggio ottenendo un feedback immediato. Oltre a questo, FilterAut calcola l'altezza del segnale e invia la nota corrispondente come una sequenza di comandi sul canale MIDI collegato al calcolatore; ogni altra applicazione che possa interpretare i comandi MIDI è in grado di interfacciarsi con FilterAut e di sincronizzarsi con il segnale analogico in ingresso.

Nell'ambito della SLMI, i moduli *CellAut* e *FilterAut* si collocano rispettivamente al Livello S/W 1 (Esecutivo/Operativo) e al Livello S/W 2 (Simbolico/Strutturale) (Camurri et al., 1989, parte I, §P1) e per quanto riguarda gli ambiti operativi specifici della SLMI, potranno essere utilizzati come strumento di ausilio alle attività di Preparazione di modelli timbrici e Analisi-sintesi suono (Camurri et al., 1989, parte I, §P2).

#### P5.15.2 Specifiche funzionali di alto livello di CellAut.

##### Diagramma funzionale.



### Descrizione funzionale in pseudocodice ADA-like.

-- Descrizione modulo CellAut

**package** CellAut **is**

**procedure** Editor di modelli

**procedure** Sintetizzatore

**end** CellAut

-- Tipi di strutture dati utilizzati dal modulo CellAut

**type** P\_Board;

--> Suond File Standard di tipo NeXT

**type** SF\_NeXT;

- -> Sound File Standard di tipo NeXT

**type** TBL;

- -> File di formato interno contenente i parametri di

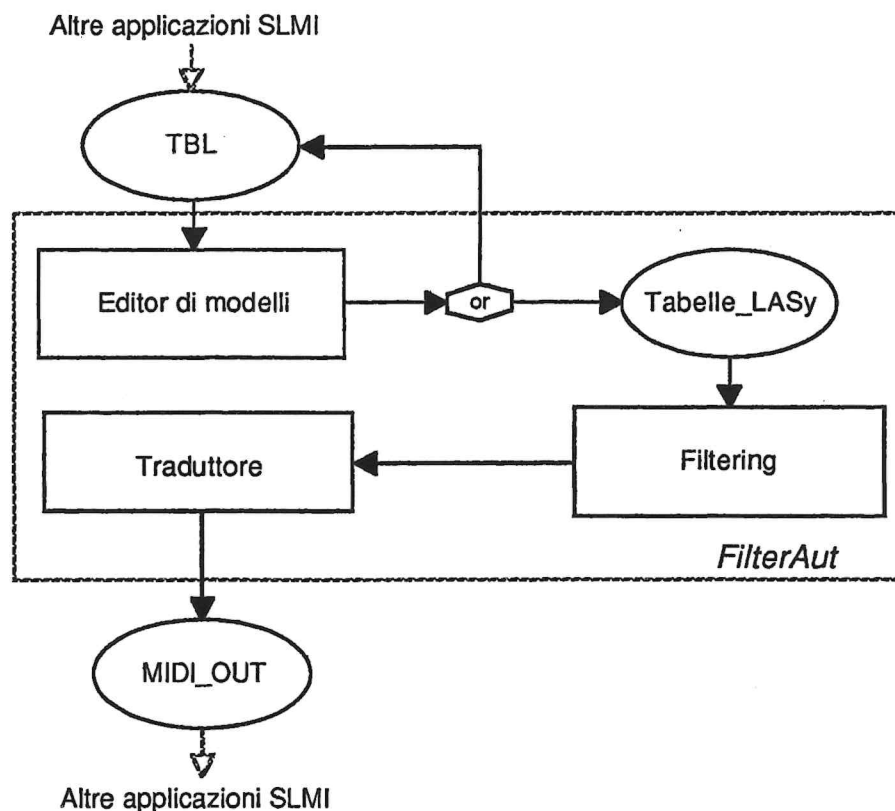
- -> sintesi

**type** Tabelle\_LASy;

--> Tabelle di valori parametrici per la sintesi LASy

### **P5.15.2      Specifiche funzionali di alto livello di FilterAut.**

#### Diagramma funzionale.





## Descrizione funzionale in pseudocodice ADA-like.

- - Descrizione modulo *FilterAut*

```
package FilterAut is
    procedure Editor di modelli
    procedure Filtering
    procedure Traduttore
end FilterAut

-- Tipi di strutture dati utilizzati dal modulo FilterAut

type MIDI_OUT;
- -> Dati standard MIDI diretti a interfaccia MIDI
type TBL;
- -> File di formato interno contenente i parametri di
- -> sintesi
type Tabelle_LASy;
--> Tabelle di valori parametrici per la sintesi LASy
```

Per quanto riguarda i dettagli implementativi e le modalità d'uso del modulo *Cellular Automata Sound Processor* si faccia riferimento al corrispondente documento operativo MUSIC (M32, 1992).

### P5.16. Modulo "Driver MIDI/DSP".

#### P5.16.1 Generalità.

Questa applicazione è stata sviluppata per consentire a qualsiasi altra applicazione che si appoggia a MMT di accedere facilmente e in tempo reale alla scheda SA. Si realizza così un veloce sistema di controllo del DSP tramite messaggi di tipo MIDI. Il *Driver MIDI/DSP* è un naturale sviluppo del *Sequencer MIDI/DSP*, nel senso che ne ha potenziato la particolarità più significativa (il controllo della scheda come un normale dispositivo MIDI) in accordo con l'atteggiamento degli sviluppatori di software commerciale, i quali si sono sempre più adeguati allo standard MMT.

Il *Driver MIDI/DSP* consente di gestire fino a 128 bank di 128 suoni ciascuno, che possono essere simultaneamente accessibili. Il bank corrente si può cambiare, sempre via MIDI, tramite un Control Message impostabile

a piacimento dall'utente, mentre il suono relativo al bank corrente è normalmente accessibile tramite un opportuno messaggio di Program Change. E' possibile leggere i Patch File generati dal *Sequencer MIDI/DSP* e le tabelle generate da *LASy* e *Waver*. Sono inoltre state implementate delle funzioni per la copia, lo spostamento e la cancellazione dei suoni per rendere più facile la gestione di un così elevato numero di suoni.

Un altro aspetto significativo è stata la messa a punto di un sistema per un test massivo in tempo reale della capacità di generazione timbrica per le sintesi TVF e WSH, le sole a garantire una certa "continuità" nell'evoluzione timbrica in dipendenza della variazione dei parametri di sintesi. Per cui sono stati costruiti dei dialoghi per scegliere quali parametri variare e con quanta intensità per poi poterne subito ascoltare l'effetto, controllando l'intensità della variazione in tempo reale tramite il movimento della Pitch Wheel di una qualsiasi tastiera MIDI. Una volta scelta una combinazione interessante, la si può automaticamente salvare come un bank di suoni, richiamabili in qualunque altro momento da messaggi di Program Change.

Nell'ambito della SLMI, il modulo *Driver 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, *Driver 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).

## P5.16.2 Specifiche funzionali di alto livello.

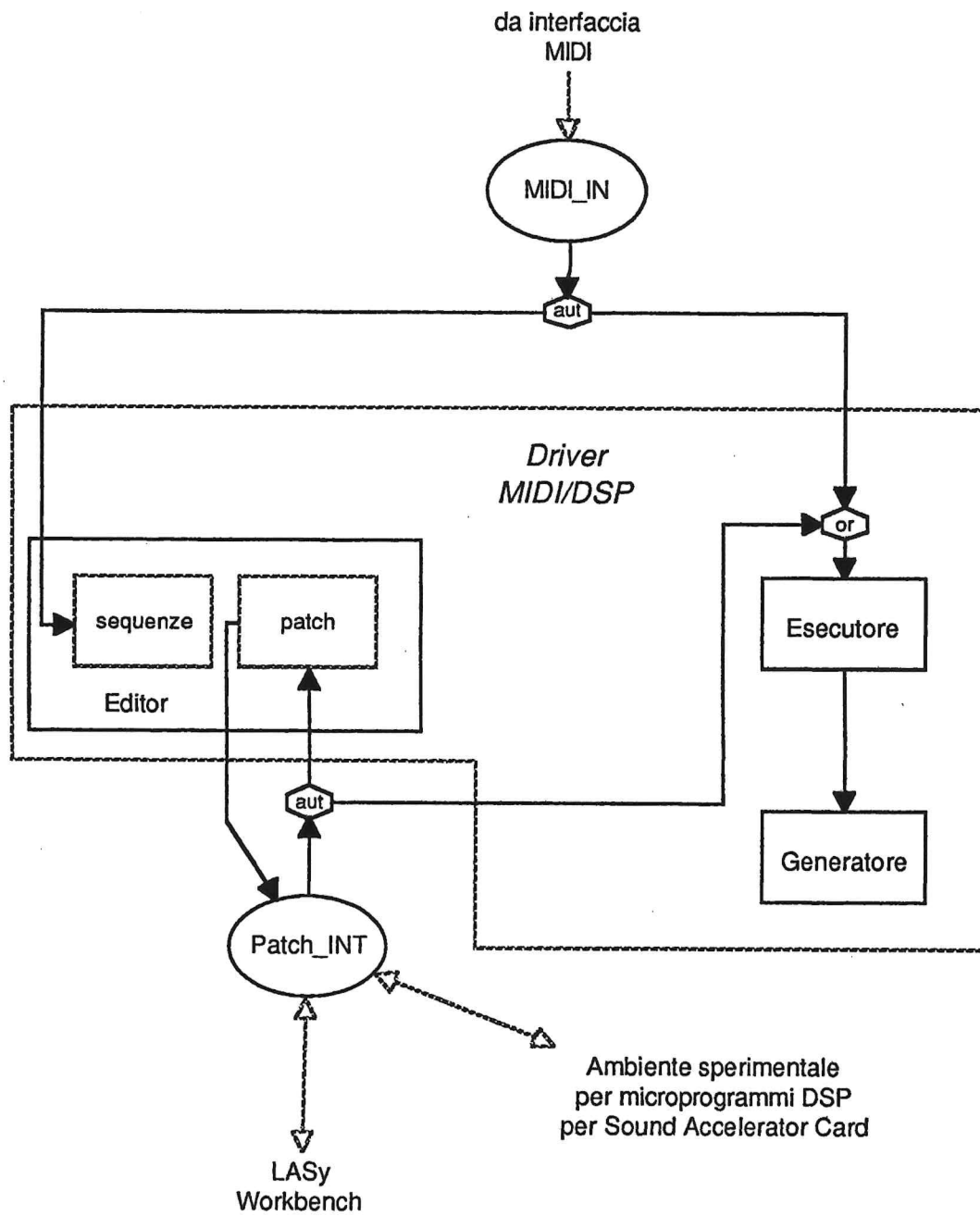
### Diagramma funzionale.

vedi pagina 64 aggiornare

### Descrizione funzionale in pseudocodice ADA-like.

-- Descrizione modulo *Driver MIDI/DSP*

```
package Driver_MIDI_DSP is
  procedure Editor;
  procedure Esecutore in tempo reale;
  procedure Generatore Suoni;
end Driver_MIDI_DSP;
```



```

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

type Patch_INT;
--> Dati patch (corrispondenze timbri <-> strumenti
--> virtuali) in formato interno
type MIDI_IN;
--> Dati standard MIDI provenienti da interfaccia MIDI MMT

```

Per quanto riguarda i dettagli implementativi e le modalità d'uso del modulo *Driver MIDI/DSP* si faccia riferimento al corrispondente documento operativo MUSIC (M26, 1992).

## **P5.17.      Interfaccia utente ipertestuale.**

### **P5.17.1      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.

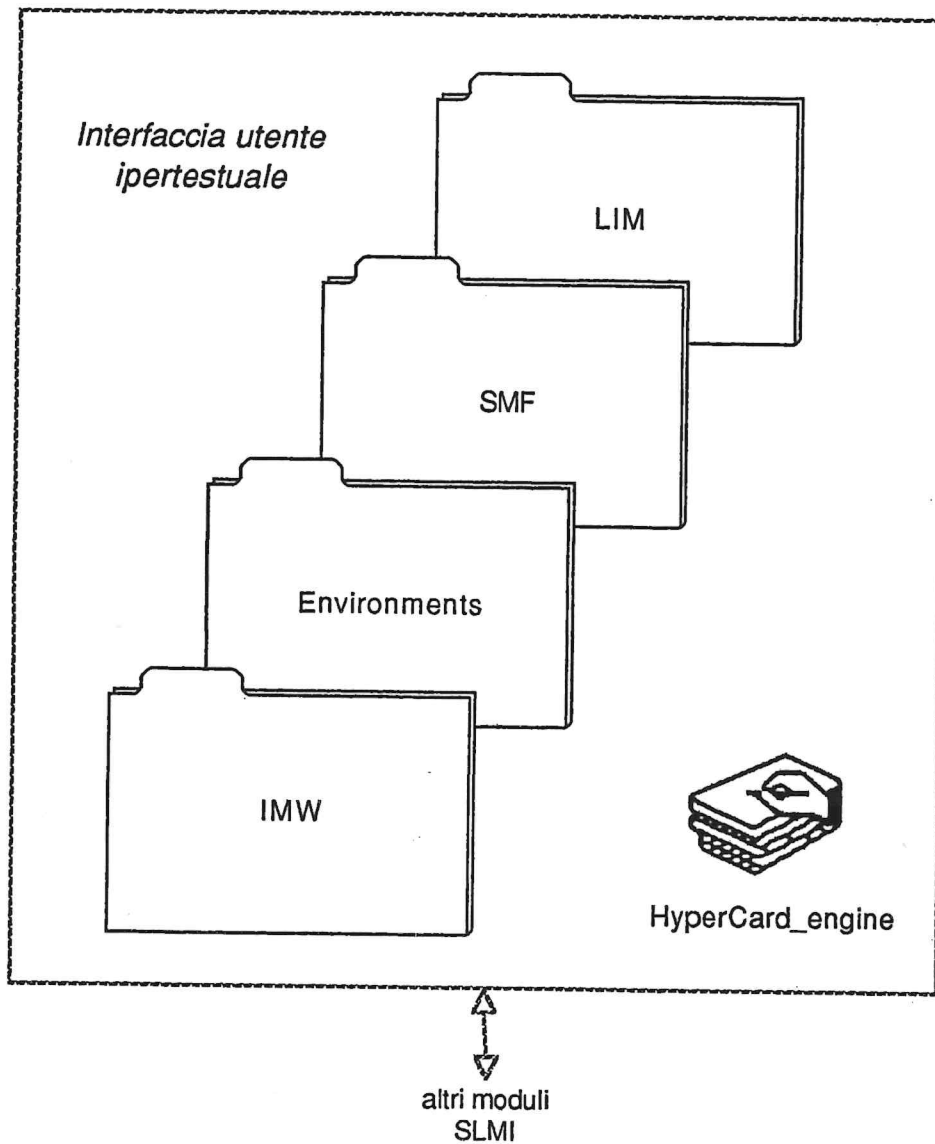
L'*interfaccia utente ipertestuale* è realizzata in ambiente Hypercard, ed è strutturata come un insieme di 4 stack interconnessi:

- 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 a 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.* Poiché tutti i moduli sperimentali e le applicazioni commerciali che compongono la SLMi possono scambiarsi informazioni MIDI sotto forma di Standard MIDI File 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 File 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).

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 MUSIC (M11, 1991).

#### **P5.18.      ModelSynth**

E' in corso la realizzazione del modulo ModelSynth, il quale permette di ricavare automaticamente un modello di Reti di Petri a partire dall'analisi effettuata da ScoreSegmenter.

A tal fine si è reso necessario un intervento di estensione per quanto riguarda i formati di output di quest'ultimo modulo.

I modelli di Reti di Petri prodotti sono a loro volta utilizzati per eventuali analisi, eseguiti e o manipolati dal modulo ScoreSynth.

#### **P6.      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. (1988). 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. (1988). 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*, Vol.14, N.4, pp.25-41, 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. (1980). Metodi di definizione di specifiche funzionali di procedure software con uso di reti di Petri. Documento n. METOD-24, PFI-CNR, Sottoprogetto P1, Obiettivo METOD.
- Haus, G., Morini, P. (1992). TEMPER: a System for Music Synthesis from Animated Tessellations, Leonardo (numero monografico su "Visual Mathematics"), *Journal of the ISAST*, Vol.25, N.3/4, pp.355-360, Pergamon Journals, San Francisco, CA.
- 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* (numero monografico su "Computer Generated Music"), Vol.24, N.7, pp.56-60.
- Haus, G., Sametti, A. (1991). "SCORESYNTH: a System for the Synthesis of Music Scores based on Petri Nets and a Music Algebra", versione estesa dell'omonimo articolo apparso su *IEEE Computer*, in "Readings in Computer Generated Music", D. Baggi Ed., pp.53-78, IEEE Computer Society Press.

- 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.
- Kung, C.H. (1989). Conceptual Modeling in the Context of software development. IEEE Trans. on Software Engineering, 15(10), 1176-1187.
- 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.
- Liu, L.C., & Horowitz, H. (1989). A Formal Model for Software Project Management. IEEE Trans. on Software Engineering, 15(10), 1280-1293.
- 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.
- Markel, J.D. & Gray, A.H. (1976), Linear Prediction of Speech, Springer-Verlag, New York.
- Mitsuhashi, 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.
- Rabiner, L.R. & Schafer, R.W. (1978). Digital Processing of Speech Signals, Prentice Hall.
- 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.
- Schoen, E., Smith, R.G., & Buchanan, B.G. (1988). Design of Knowledge-based Systems with a Knowledge-based Assistant. IEEE Trans. on Software Engineering, 14(12), 1771-1790.
- 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.
- Sommerville, I. (1989). Software Engineering. Addison-Wesley Publ.
- M/01: A. Camurri, G. Haus, A. Stiglitz, R. Zaccaria: "LRC C4 MUSIC: orientamenti generali, definizione degli intenti e premessa alle specifiche funzionali", CNR-PFI2, novembre 1989.

- M/02: A. Camurri, G. Haus, A. Stiglitz, R. Zaccaria: "LRC C4 MUSIC: architettura, specifiche funzionali di alto livello e standard di documentazione della STAZIONE DI LAVORO MUSICALE INTELLIGENTE", CNR-PFI2, novembre 1989.
- M/03: A. Sametti, S. Scolaro: Guida operativa del modulo "ScoreSynth", CNR-PFI2, marzo 1991.
- M/04: P. Bettitoni: Note tecniche del modulo "Routines di lettura/scrittura di Standard MIDI File 1.0", CNR-PFI2, marzo 1991.
- M/05: A. Ballista, E. Casali, J. Chareyron: Guida operativa del modulo "Sequencer MIDI/DSP", CNR-PFI2, marzo 1991.
- M/06: AA.VV. (A. Stiglitz editor): "Specifiche funzionali di alto livello dei moduli della STAZIONE DI LAVORO MUSICALE INTELLIGENTE - Unità Operativa dell'Università degli Studi di Milano", CNR-PFI2, marzo 1991.
- M/07: A. Sametti, S. Scolaro: Note tecniche del modulo "ScoreSynth", CNR-PFI2, giugno 1991.
- M/08: L. Lazzini (M. E. Caputi editor): Guida operativa del modulo "MusSer", CNR-PFI2, giugno 1991.
- M/09: A. Ballista, E. Casali, J. Chareyron: Note tecniche del modulo "Sequencer MIDI/DSP", CNR-PFI2, giugno 1991.
- M/10: E. Casali, J. Chareyron: Note tecniche del modulo "Libreria microprogrammi per Sequencer MIDI-DSP e Sound Accelerator Card (TVF, Automi Cellulari, Waveshaping)", CNR-PFI2, giugno 1991.
- M/11: A. Monizza: Guida operativa dell'interfaccia utente ipertestuale della "Stazione di lavoro musicale intelligente", CNR-PFI2, giugno 1991.
- M/12: A. Monizza: Note tecniche dell'interfaccia utente ipertestuale della "Stazione di lavoro musicale intelligente", CNR-PFI2, giugno 1991.
- M/13: G. Penocchio: Guida operativa del modulo "SampleBuster", CNR-PFI2, giugno 1991.
- M/14: G. Penocchio: Note tecniche del modulo "SampleBuster", CNR-PFI2, giugno 1991.
- M/15: E. Casali: Guida operativa del modulo "Ambiente sperimentale per microprogrammi DSP per Sound Accelerator Card (TVF, Waveshaping, Wavelet)", CNR-PFI2, giugno 1991.
- M/16: E. Casali: Note tecniche del modulo "Ambiente sperimentale per microprogrammi DSP per Sound Accelerator Card (TVF, Waveshaping, Wavelet)", CNR-PFI2, giugno 1991.
- M/17: E. Casali: Note tecniche del modulo "Libreria microprogrammi per Ambiente sperimentale DSP e Sound Accelerator Card (TVF, Waveshaping, Wavelet)", CNR-PFI2, giugno 1991.
- M/18: F. Lonati: Guida operativa del modulo "ScoreSegmenter", CNR-PFI2, giugno 1991.
- M/19: F. Lonati: Note tecniche del modulo "ScoreSegmenter", CNR-PFI2, giugno 1991.
- M/20: M. Bastardi, P. Crippa, A. Rodriguez (M. E. Caputi editor): Guida operativa del modulo "Traslitteratore", CNR-PFI2, giugno 1991.

- M/26: A. Ballista: Guida operativa del modulo "Driver MIDI/DSP per schede DSP Digidesign", CNR-PFI2, ottobre 1992.
- M/27: A. Ballista: Note tecniche del modulo "Driver MIDI/DSP per schede DSP Digidesign", CNR-PFI2, ottobre 1992.
- M/28: I. Pighi: Guida operativa del modulo "TimbreLab", CNR-PFI2, ottobre 1992.
- M/29: I. Pighi: Note tecniche del modulo "TimbreLab", CNR-PFI2, ottobre 1992.
- M/30: J. Chareyron: Guida operativa del modulo "LASy Workbench", CNR-PFI2, ottobre 1992.
- M/31: J. Chareyron: Note tecniche del modulo "LASy Workbench", CNR-PFI2, ottobre 1992.
- M/32: D. Rizzi: Guida operativa del modulo "Cellular Automata Sound Processor", CNR-PFI2, dicembre 1992.
- M/33: D. Rizzi: Note tecniche del modulo "Cellular Automata Sound Processor", CNR-PFI2, dicembre 1992.
- M/34: P. Morini, D. Tanzi: Guida operativa del modulo "TEMPER", CNR-PFI2, dicembre 1992.
- M/35: E. Catalano: Guida operativa del modulo "VoiceLab", CNR-PFI2, dicembre 1992.
- M/36: E. Catalano: Note tecniche del modulo "VoiceLab", CNR-PFI2, dicembre 1992.
- M/37: A. Sametti: Guida operativa del modulo "ScoreSynth 3.0", CNR-PFI2, dicembre 1992.
- M/38: A. Sametti: Note tecniche del modulo "ScoreSynth 3.0", CNR-PFI2, dicembre 1992.
- M/39: F. Madeddu: Guida operativa del modulo "Functional Performer", CNR-PFI2, dicembre 1992.
- M/40: F. Madeddu: Note tecniche del modulo "Functional Performer", CNR-PFI2, dicembre 1992.

