UNIVERSITA' DEGLI STUDI DI
NAPOLI FEDERICO II
**Facoltà di Ingegneria**
**Corso di Studi in Ingegneria Informatica**

tesi di laurea

# Modelli per il Reverse Engineering delle applicazioni “Flash”

Anno Accademico 2005/2006

**Relatore**
Ch.mo prof. Porfirio Tramontana

**candidato**
Vincenzo Guadagno
Matr. 041/003408

UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II

# Presentazione – Outline

- Il Macromedia Flash, insieme al linguaggio ActionScript in esso contenuto, è una delle tecnologie maggiormente in ascesa in ambito tecnologico negli ultimi anni.

- Grazie alla sua interfaccia intuitiva ed alla sua semplicità di utilizzo, viene usato spesso anche da chi non è esperto di linguaggi di programmazione e di scripting.

- Non sono presenti attualmente in letteratura scientifica molti lavori riguardanti l'analisi e l'astrazione delle applicazioni "Flash". Ed è questo il punto cruciale dal quale si è partiti per lo sviluppo dell'elaborato che si sta per presentare.

Si vuole infatti:

✓ Analizzare ed astrarre le "Applicazioni Flash" ad un livello di più alto di quello dell'utilizzatore.

✓ Consentire la catalogazione degli elementi in esse contenuti.

✓ Permettere eventuali reingegnerizzazioni delle applicazioni stesse.

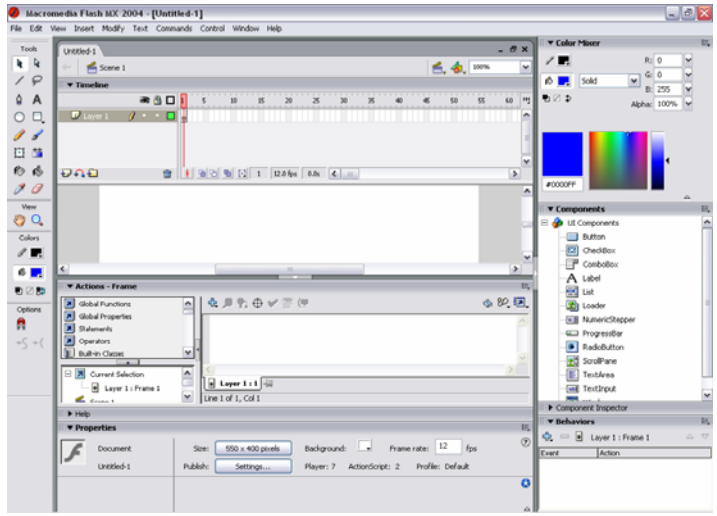
UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II

# Macromedia Flash per il web

Il “Macromedia Flash” attualmente viene utilizzato soprattutto nell'ambito della **costruzione di pagine web**.

Sempre più spesso, infatti, durante la navigazione, ci si imbatte in siti:

- interamente realizzati tramite questa tecnologia;
- che contengano come “contenuti speciali” delle animazioni Flash.

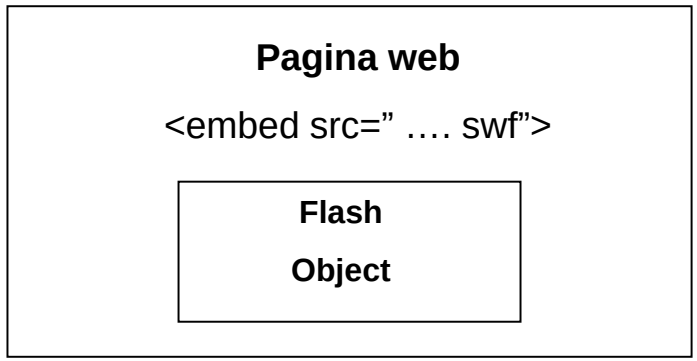



➢ La crescita in questo campo è stata così rapida che tutti i principali browser hanno installati al proprio interno dei **plug-in** in grado di leggere le animazioni inserite nelle pagine web.

UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II

# Letteratura delle applicazioni Flash

- Nonostante l'enorme sviluppo, non sono presenti ad oggi, documenti che riconoscano le applicazioni "Flash" quali oggetti con una propria autonomia ed una loro struttura interna.
- Nella letteratura esistente si fa riferimento ad un'applicazione Flash come **un semplice oggetto** contenuto in una pagina web.




- Questa situazione risulta un handicap per una tecnologia in forte ascesa, che vive indipendentemente dal contesto in cui la si pone e merita, senz'altro, un'analisi più accurata ed approfondita.

UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II

# Contesto delle web applications (1)

## Il modello di Conallen

- Il modello UML che si vuole realizzare colloca le F.A. in un contesto ben delineato nel campo delle web applications, detto “modello di Conallen”.

- Nella pubblicazione di questa teoria, Conallen ha utilizzato alcuni meccanismi di estensione di UML al fine di poter avere a disposizione strumenti per modellare i vari componenti relativi alle pagine web, i collegamenti tra esse ed il loro contenuto dinamico.

Ciò che ha proposto è riportato di seguito:

**Classi:**

Pagina Client

Pagina Server

Form

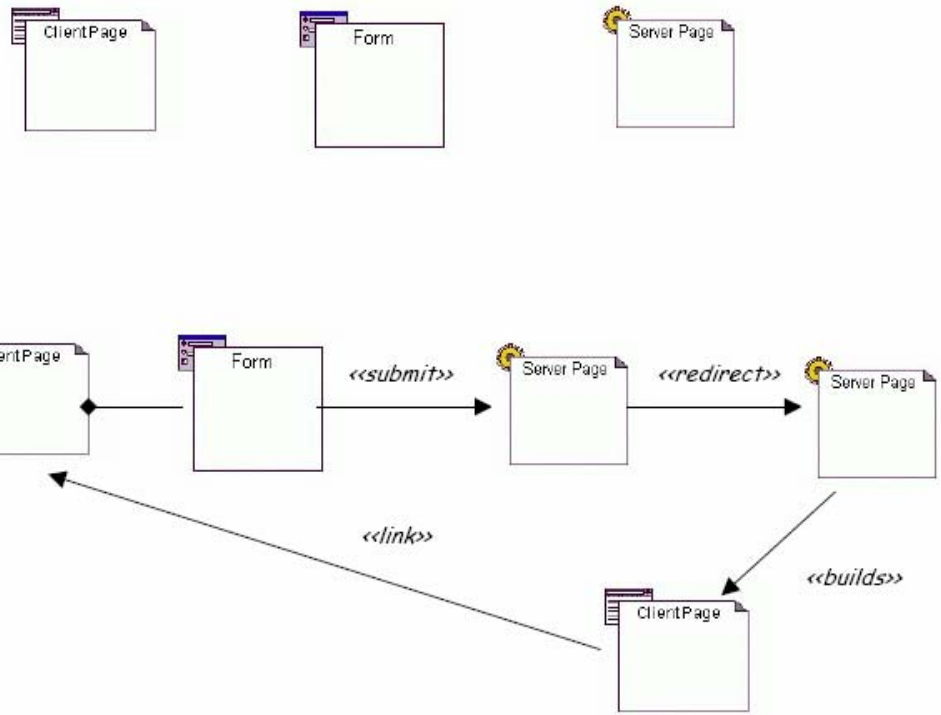
**Associazioni:**

Link

Submit

Builds

Redirect

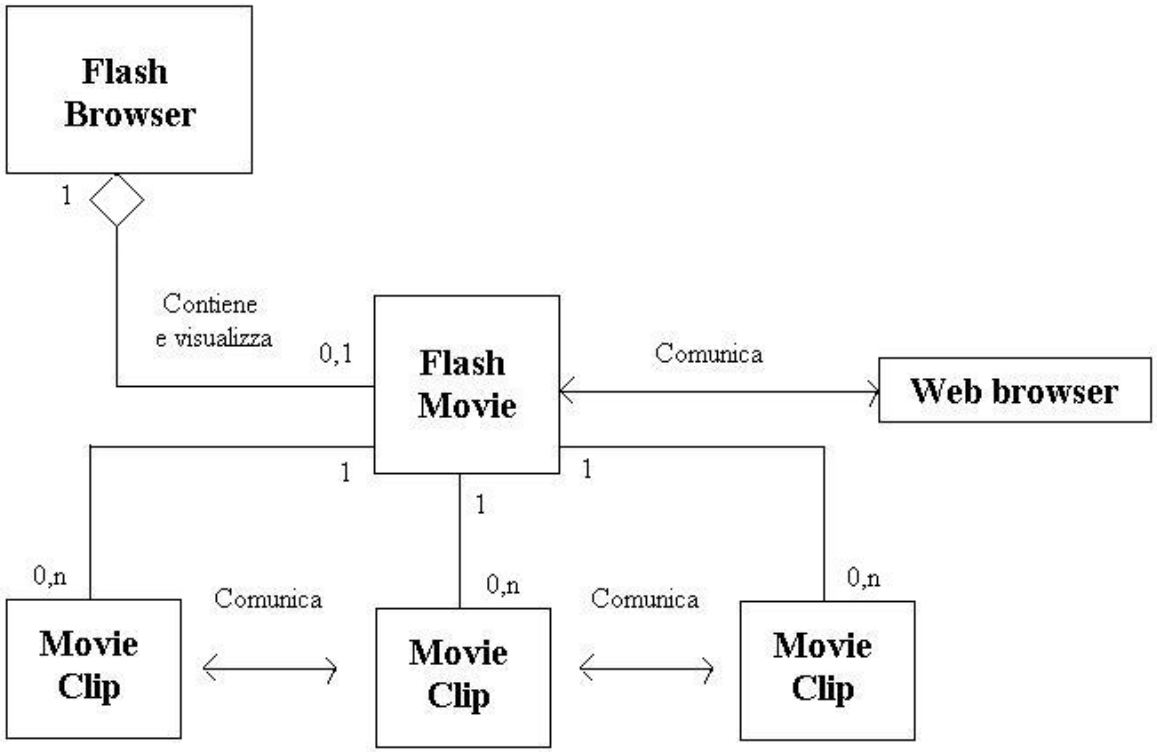


# Contesto delle web applications (2)

## UML for Flash

Una sfida è stata da qualche anno lanciata da alcuni ricercatori, i quali, sulla base del modello di Conallen, hanno pensato di estendere il tutto alle applicazioni Macromedia Flash.

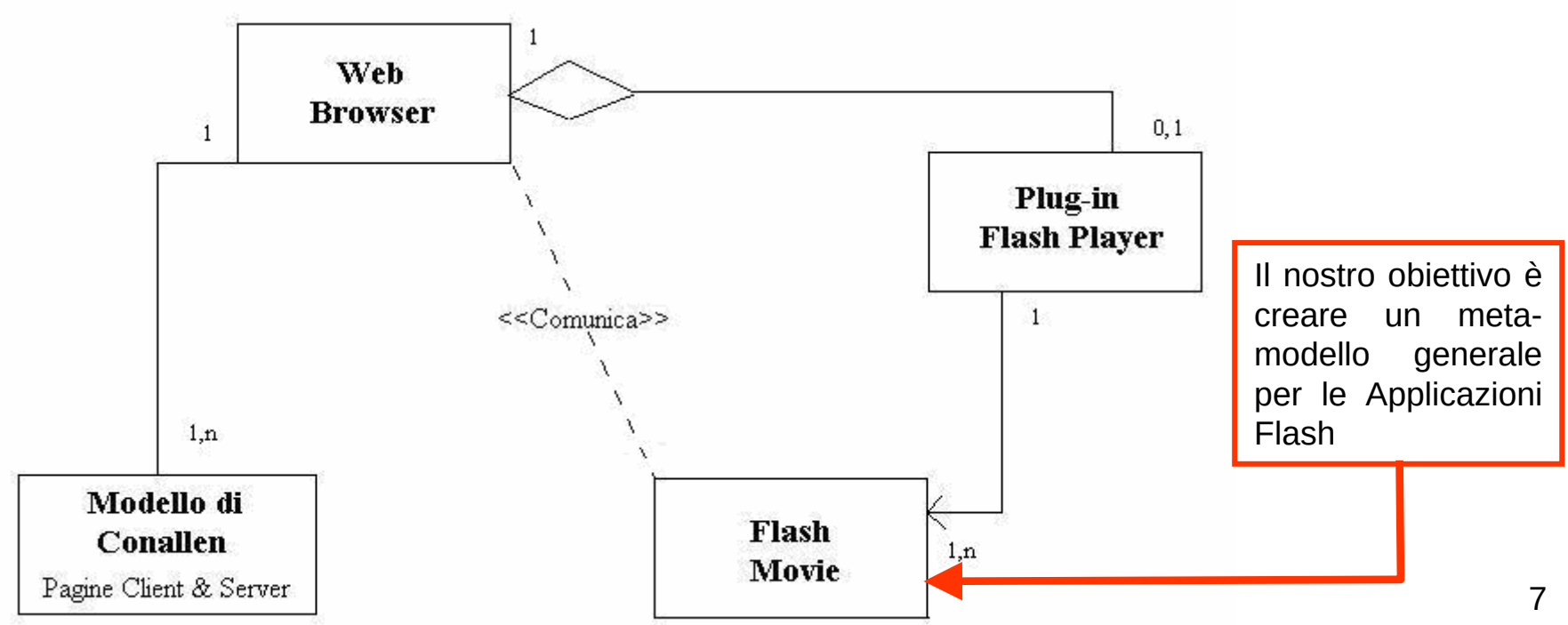
- Il Flash Player è inteso come un Browser aggiuntivo che consente la visualizzazione dell'animazione.
- La Flash Movie è l'animazione vera e propria, che comunica con il classico Web Browser e possiede all'interno una serie di "Movie Clip", anche loro in grado di comunicare eventualmente tra loro.



# Contesto delle web applications (3)

Il precedente tentativo di estendere il modello di Conallen alle applicazioni Flash non è mai stato portato a termine, poiché è risultato difficile creare un modello per le applicazioni ed interfacciarlo effettivamente.

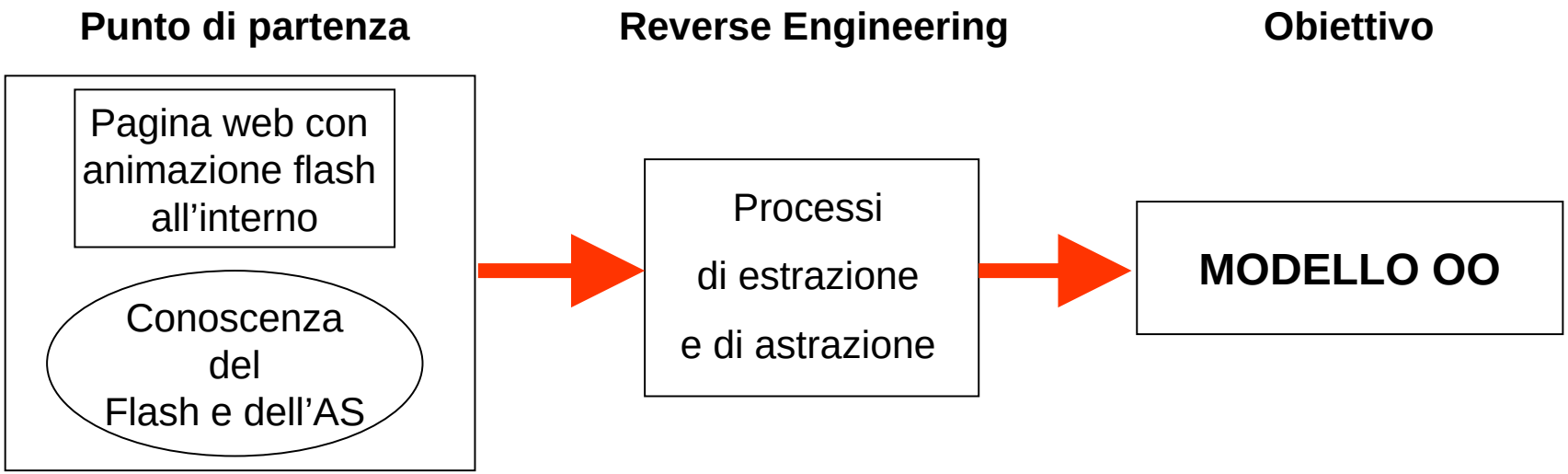
- Nel precedente modello è mostrata solo la parte dinamica di un'applicazione, collegando solo Movie Clips alla Flash Movie
- La divisione tra Flash player e browser web non ha senso.



UNIVERSITA' DEGLI STUDI DI NAPOLI FEDERICO II

# Obiettivo

- Ciò che si vuole realizzare è un approfondimento di tutto quanto legato alla realtà delle applicazioni Flash, che abbia, come fine ultimo, la realizzazione di un **modello Orientato agli Oggetti (OO) di più alto livello**, il quale evidenzi i legami tra gli elementi di una Flash Application e qualsiasi altro oggetto esterno all'applicazione.



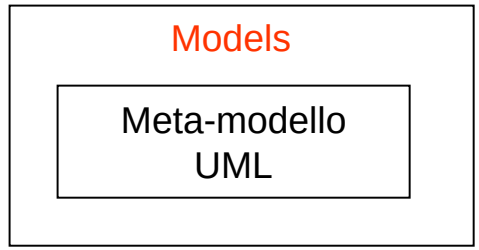
UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II

# Metodologia adottata

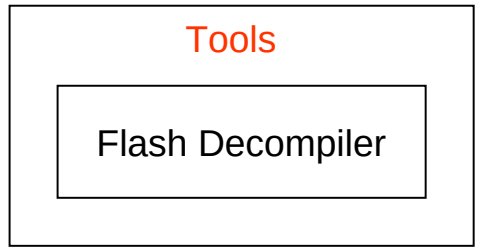
La strategia che verrà adottata per il raggiungimento degli obiettivi è quella del **Reverse Engineering**, secondo il paradigma **Goals – Models – Tools**.

Goals: Si fissano gli obiettivi da raggiungere mediante RE.


Models: Si fissa una strategia e si decide effettivamente quali documenti realizzare.




Tools: Si producono gli strumenti necessari per perseguire gli obiettivi.

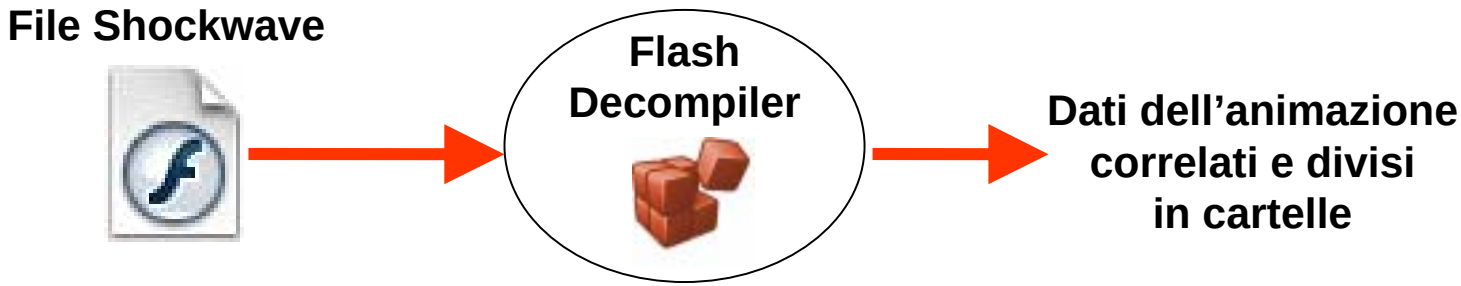


UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II

# Prima fase: estrazione dei dati

Per giungere al meta-modello proposto, la prima fase è quella relativa all'estrazione delle informazioni da una generica applicazione Flash.

Per tale processo si è usato il **Flash Decompiler**:

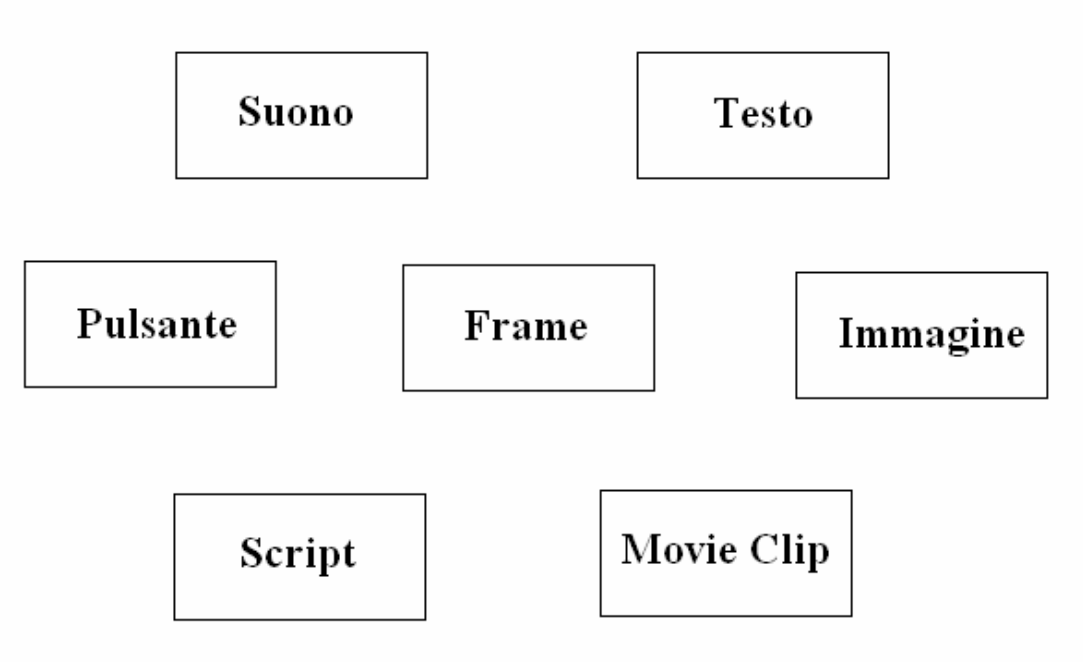



Tra le cartelle più significative che vengono restituite dal decompiler annoveriamo:

- TEXT
- BUTTON
- SCRIPT
- FRAME
- SHAPE
- IMAGE
- SPRITE
- SOUND

# Seconda fase: Astrazione del meta-modello (1)

- Dalle informazioni estratte dal Flash Decompiler è possibile ricavare che una generica applicazione Flash è costituita dai seguenti elementi:

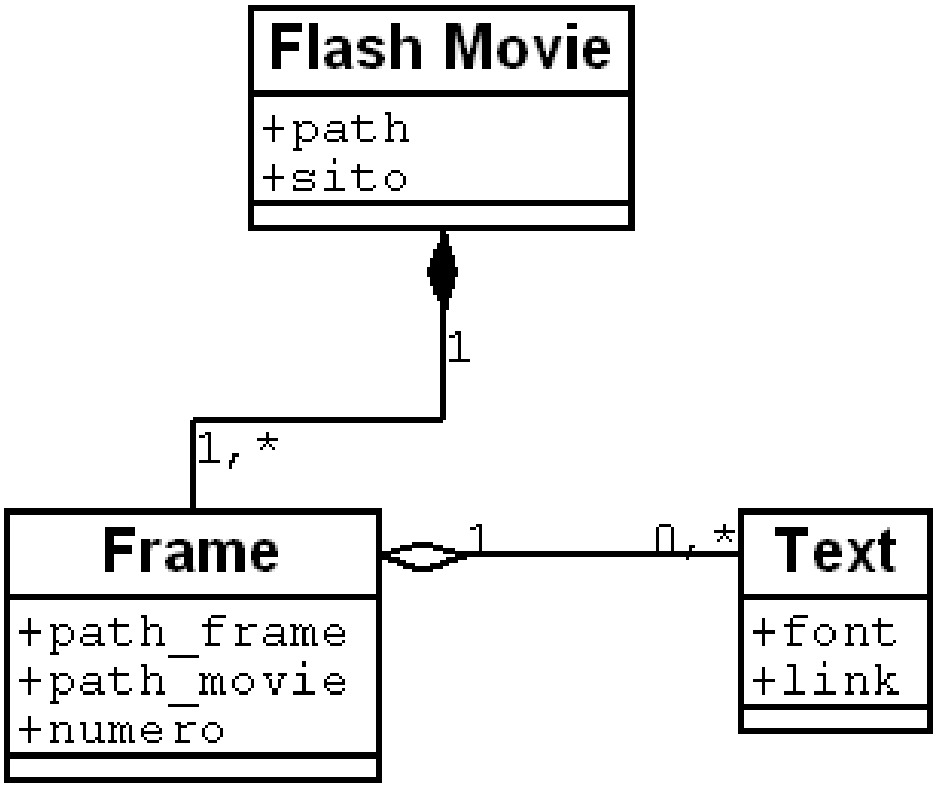



- Per ognuno di tali elementi sarà costruita una classe UML e verranno ricercate le relazioni tra le varie classi, in modo da ottenere un **Class Diagram** generale rappresentante l'animazione Flash nel suo complesso.

UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II

# Seconda fase: Astrazione del meta-modello (2)

**Prime tre classi individuate**

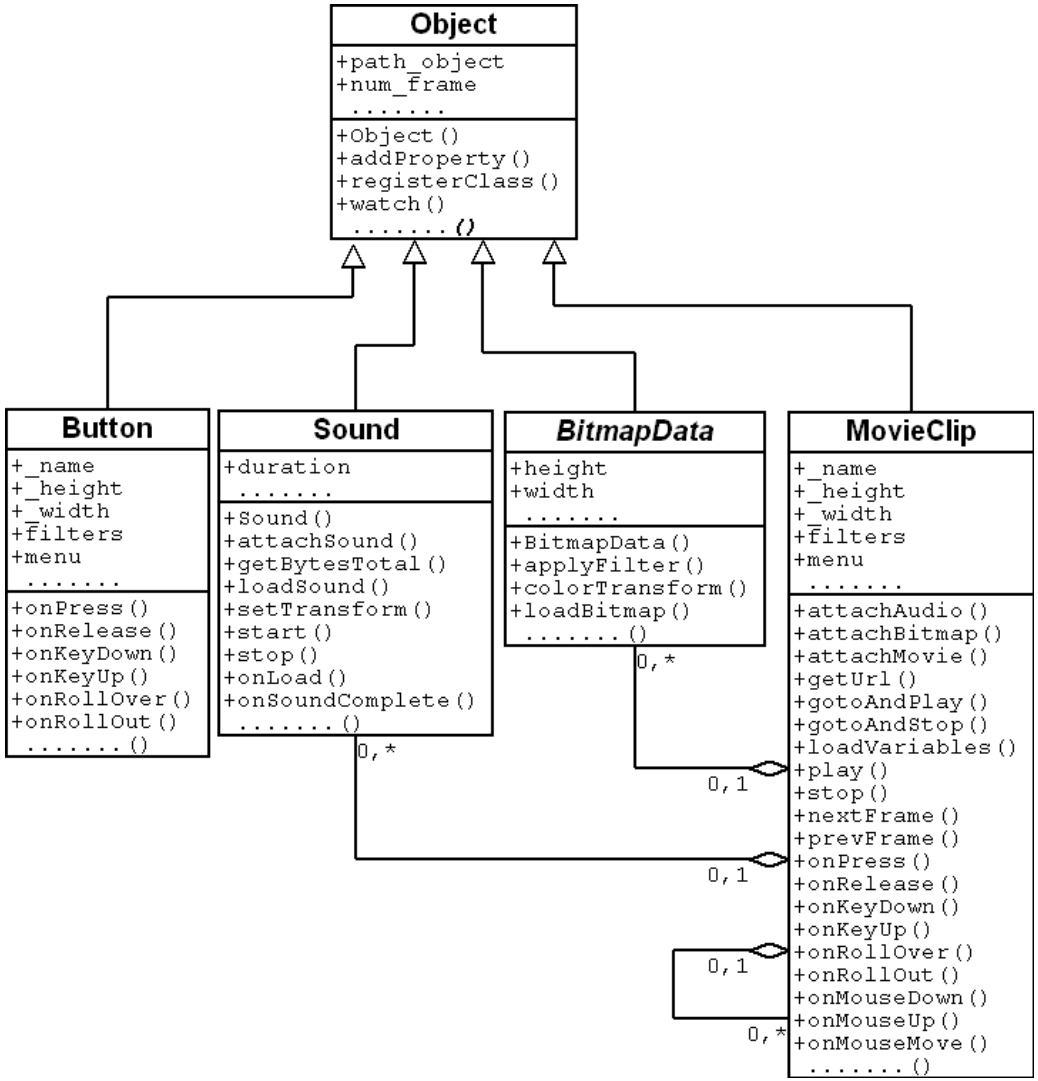



- ✓ La classe **Flash Movie** rappresenta l'animazione nel suo complesso.
- ✓ La classe **Frame** rappresenta il generico fotogramma, la "fotografia" dell'animazione in un preciso istante.
- ✓ La classe **Text** contiene testi statici che è possibile apporre allo stage di lavoro.

UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II

# Seconda fase: Astrazione del meta-modello (3)

## La gerarchia della classe Object




- Il linguaggio ActionScript definisce una serie di classi, dette ActionScript classes, ognuna delle quali rappresenta una tipologia di elemento (o effetto) che possiamo ritrovare nell'applicazione.

- Possiamo astrarre da una classe ActionScript padre (la classe **Object**), una serie di classi figlie, ognuna contenente attributi e metodi specifici.

- Per ogni classe ActionScript definiremo una classe UML.

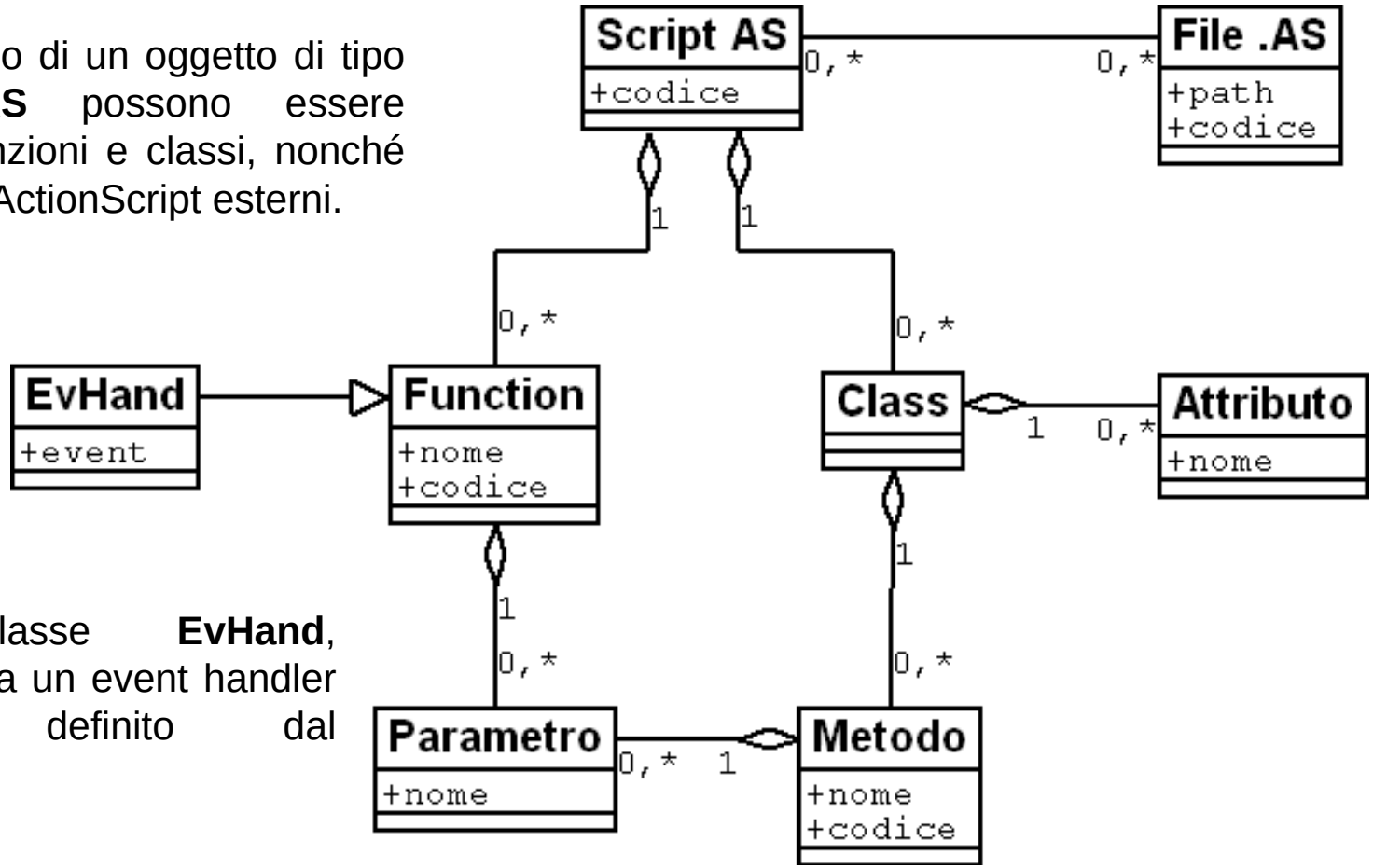
UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II

# Seconda fase: Astrazione del meta-modello (4)

## La classe Script AS

E' possibile associare le azioni più disparate ad un qualunque oggetto dell'animazione, raccolte dal decompilatore in un unico script.

✓ All'interno di un oggetto di tipo **Script AS** possono essere definite funzioni e classi, nonché inclusi file ActionScript esterni.

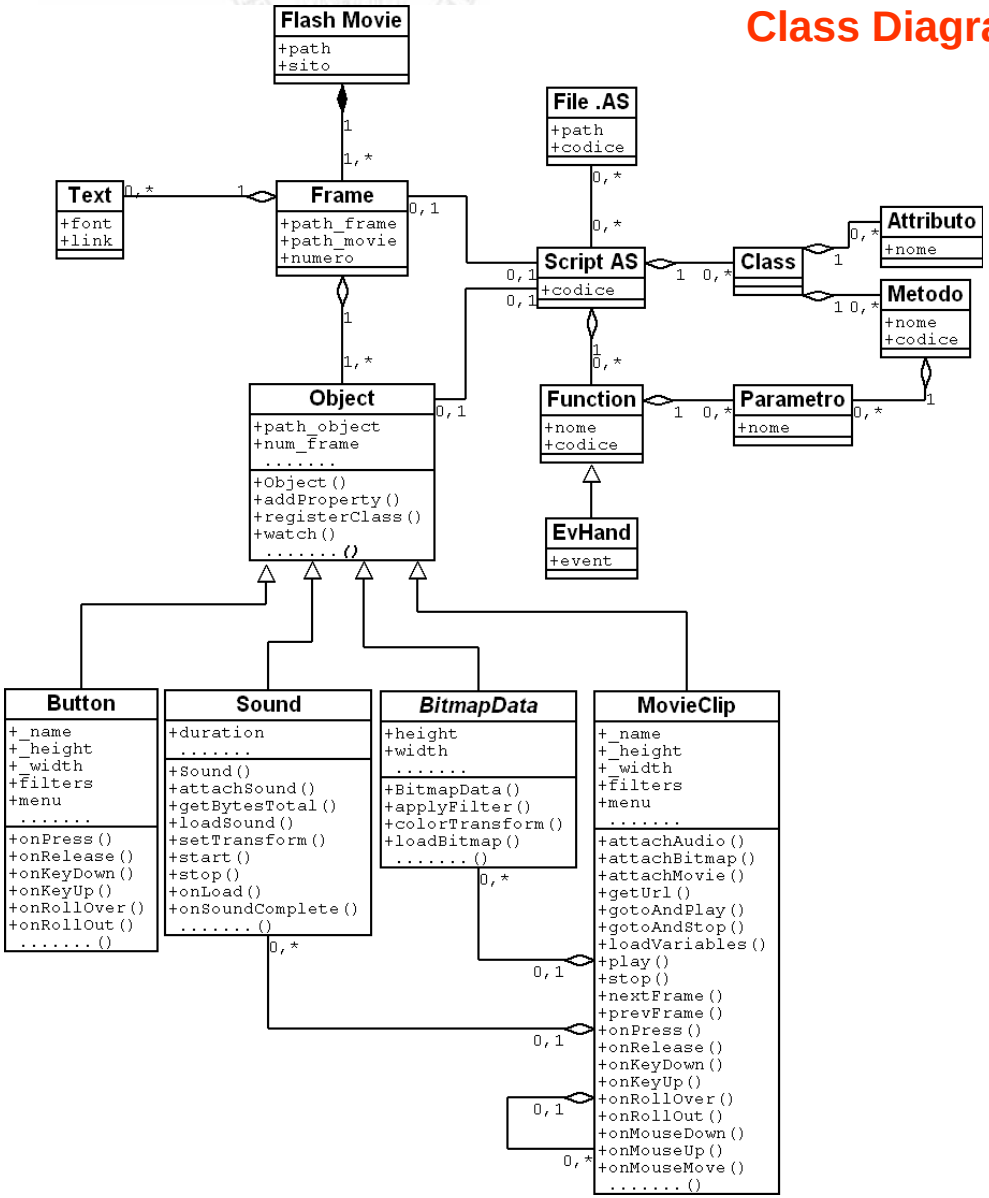



✓ La classe **EvHand**, rappresenta un event handler method definito dal progettista

UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II

# Seconda fase: Astrazione del meta-modello (5)

## Class Diagram

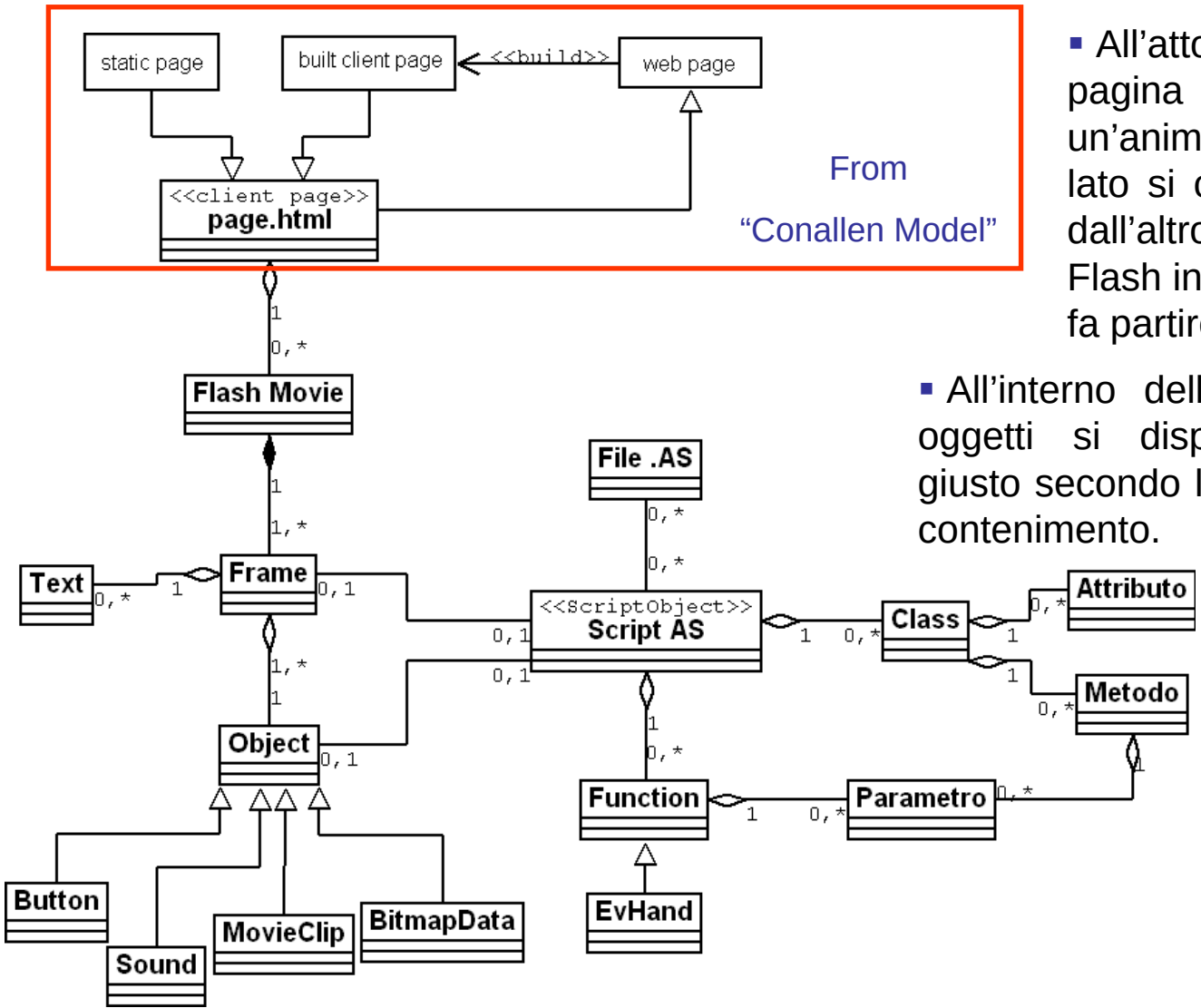



Il Class Diagram generale di un'applicazione Flash si ottiene mettendo insieme i diagrammi presentati nelle precedenti slides secondo le seguenti considerazioni.

✓ Non è possibile che esistano un pulsante, un movie clip, o un'immagine senza almeno un frame che li contenga.

✓ Ad ogni oggetto della classe Frame o di qualunque altra classe ActionScript è possibile associare uno script e, quindi, un'istanza della classe **Script AS**.

# Estensione proposta del Modello di Conallen

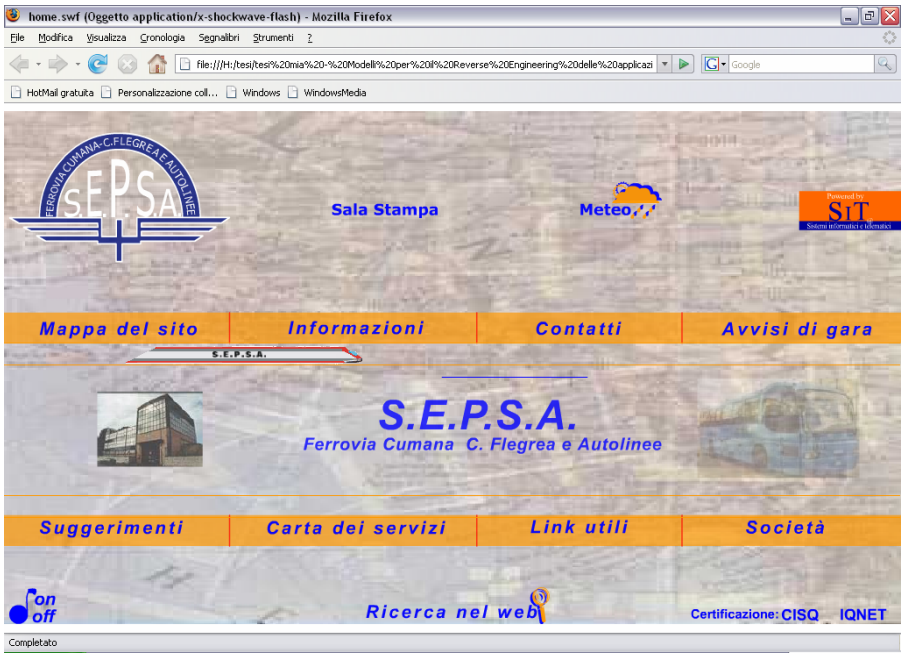



- All'atto del lancio di una pagina web contenente un'animazione Flash, da un lato si crea la pagina web, dall'altro, tramite il plug-in Flash interno del browser, si fa partire l'animazione.
- All'interno dell'animazione tutti gli oggetti si dispongono nel modo giusto secondo le relative relazioni di contenimento.

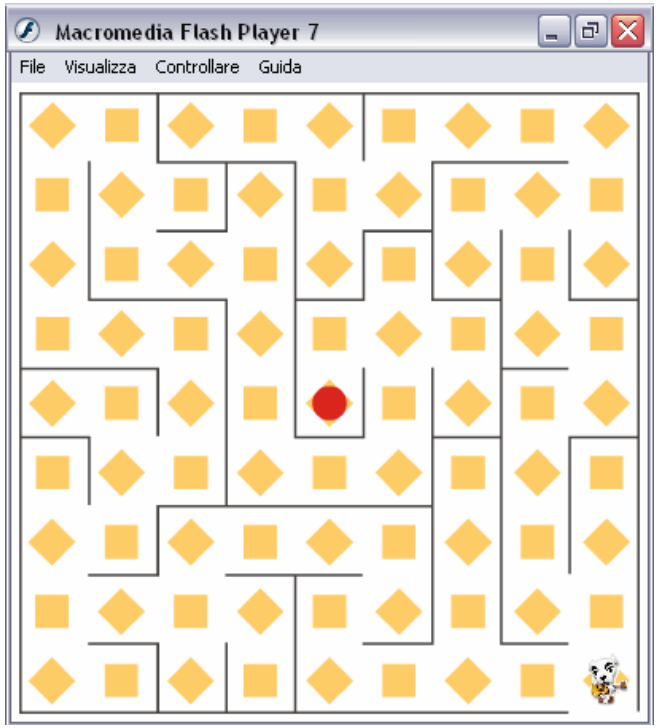
UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II

# Casi di studio

Si è applicato il meta-modello elaborato allo studio di due specifiche applicazioni Flash:




Una **pagina web** completamente realizzata in Flash, quale la home page del sito www.sepsa.it



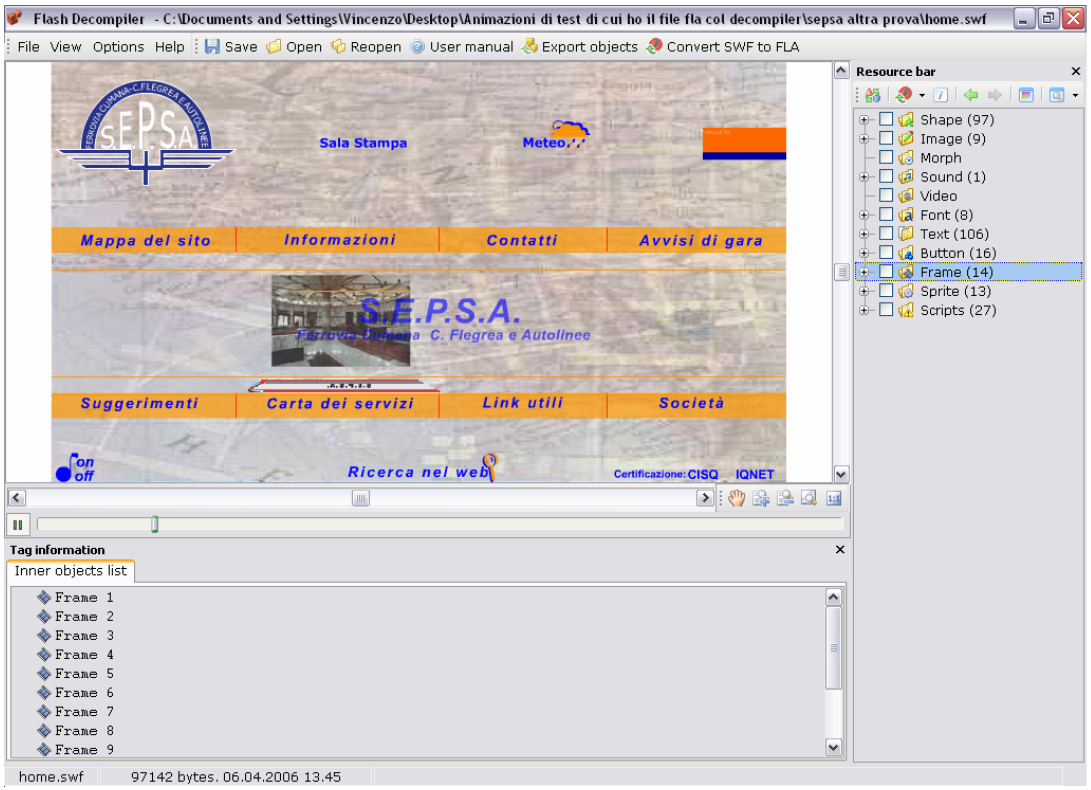

Un semplice **videogioco**

UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II

# Un primo caso di studio (1)

## Estrazione delle informazioni

Applichiamo il Flash Decompiler al sito **www.sepsa.it**.




Il decompilatore individua:

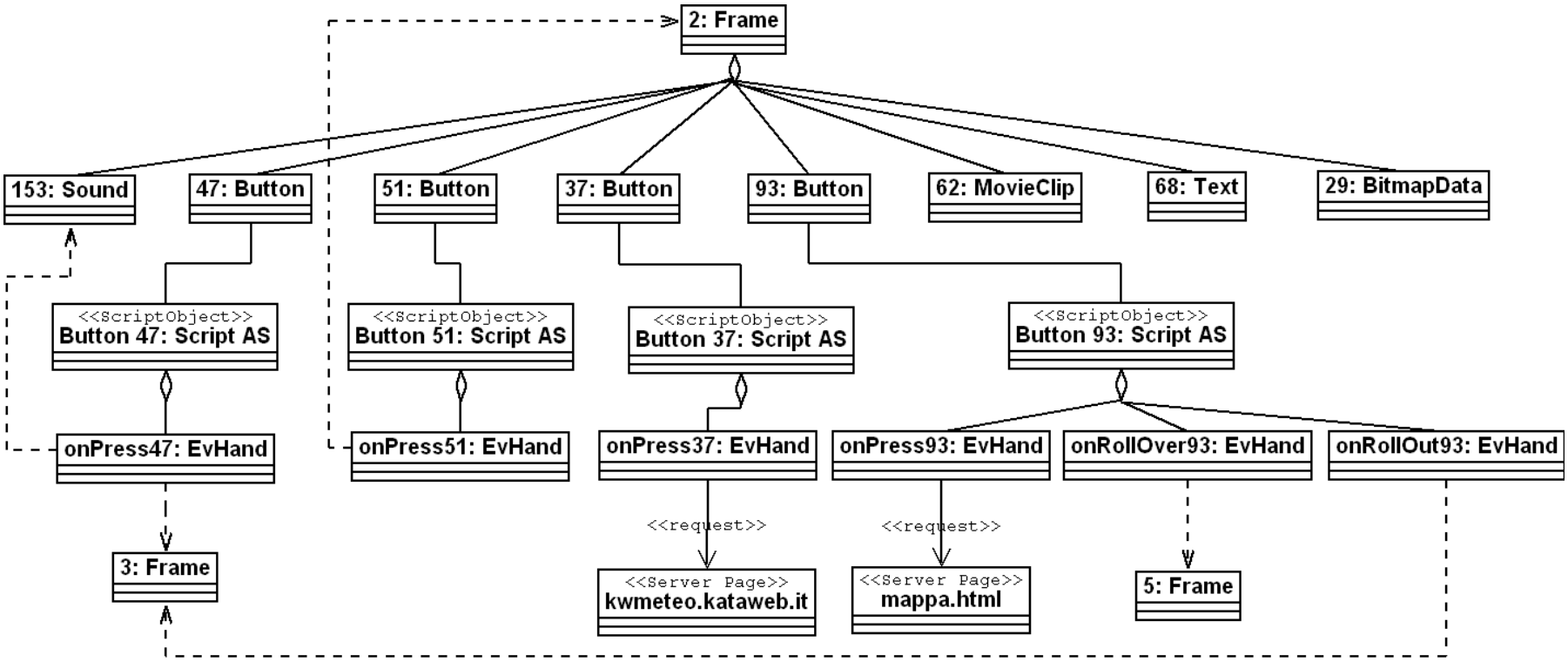
- 97 oggetti di tipo Shape;
- 9 immagini;
- nessun morph o video;
- 1 oggetto di tipo Sound;
- 106 oggetti di tipo Text;
- 16 Buttons;
- 14 Frames;
- 13 Sprites;
- 27 oggetti di tipo Script.

UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II

# Un primo caso di studio (2)

## Astrazione Object Diagrams

L'animazione esaminata è composta da quattordici frames, per ognuno dei quali è stato costruito un Object Diagram di dettaglio, come quello presentato in figura, relativo al Frame 2

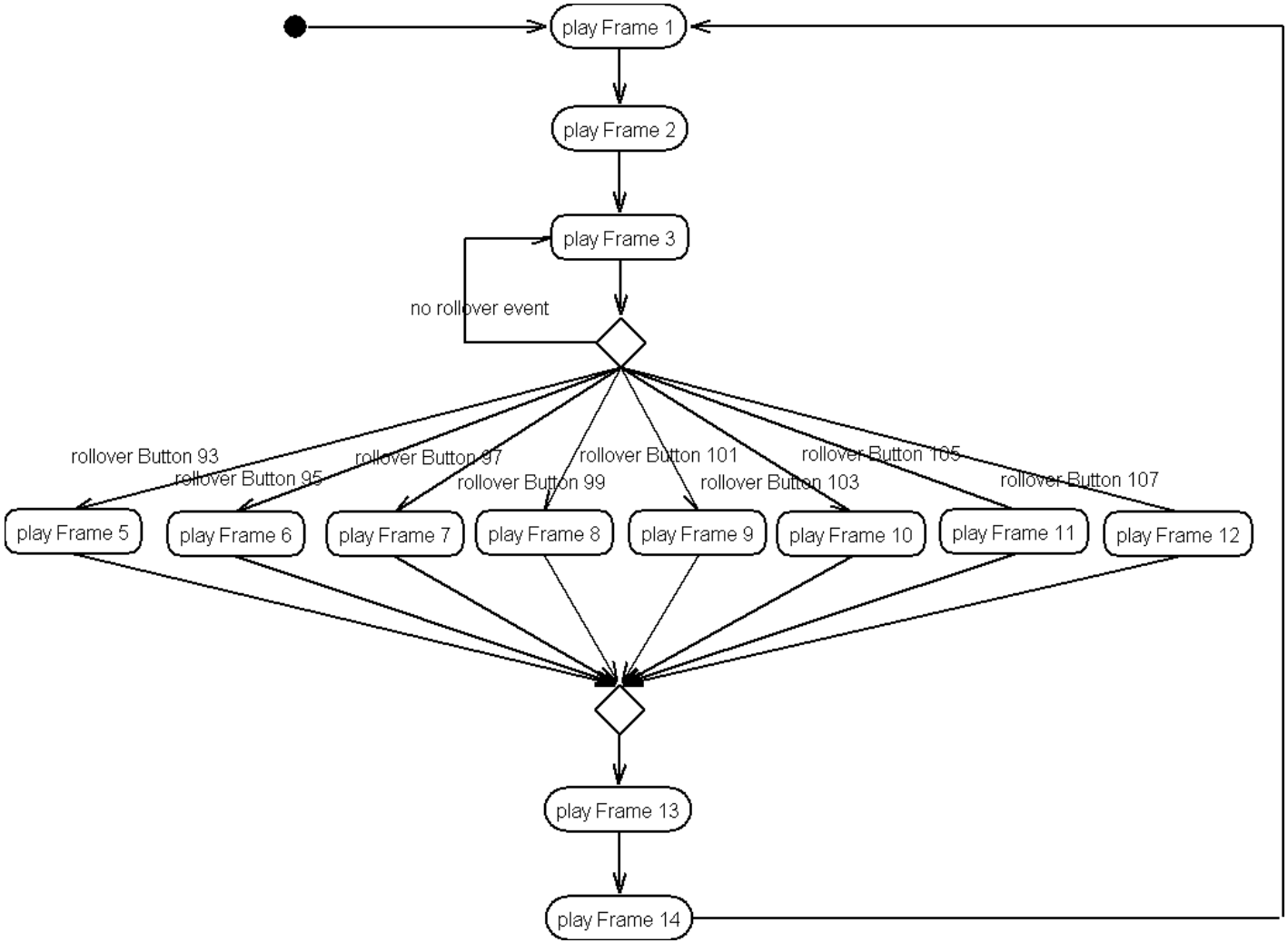


UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II

# Un primo caso di studio (3)

## Astrazione Activity Diagram

Nella figura successiva è mostrata in maggiore dettaglio la sequenza nella riproduzione dei frames.



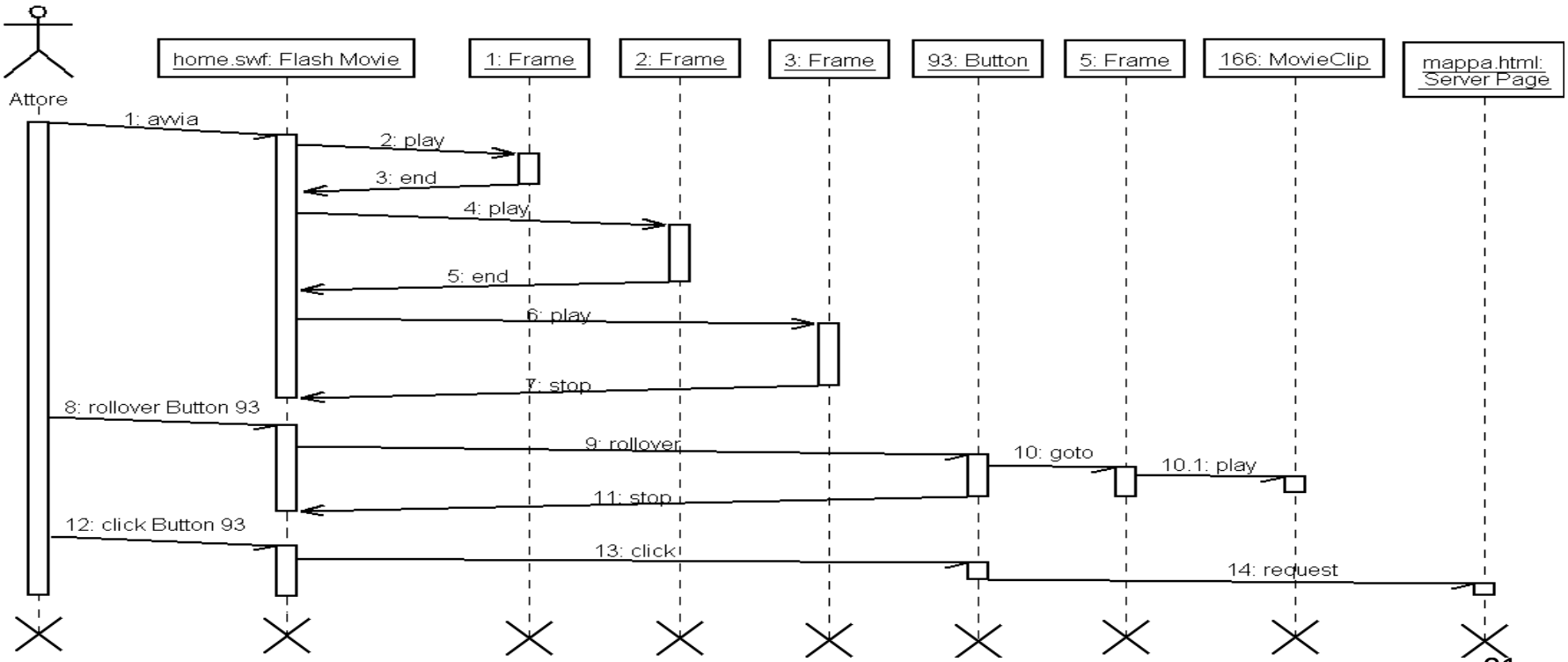
UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II

# Un primo caso di studio (4)

## Astrazione Sequence Diagrams

Mediante i Sequence Diagrams si vuole mostrare come può cambiare lo scenario temporale in funzione di ciò che il progettista decide sulla base di due fattori:

1. come eventualmente egli stesso costruisce l'animazione;
2. come l'utente può comandarla nel caso in cui gli venga data libertà in questo senso.





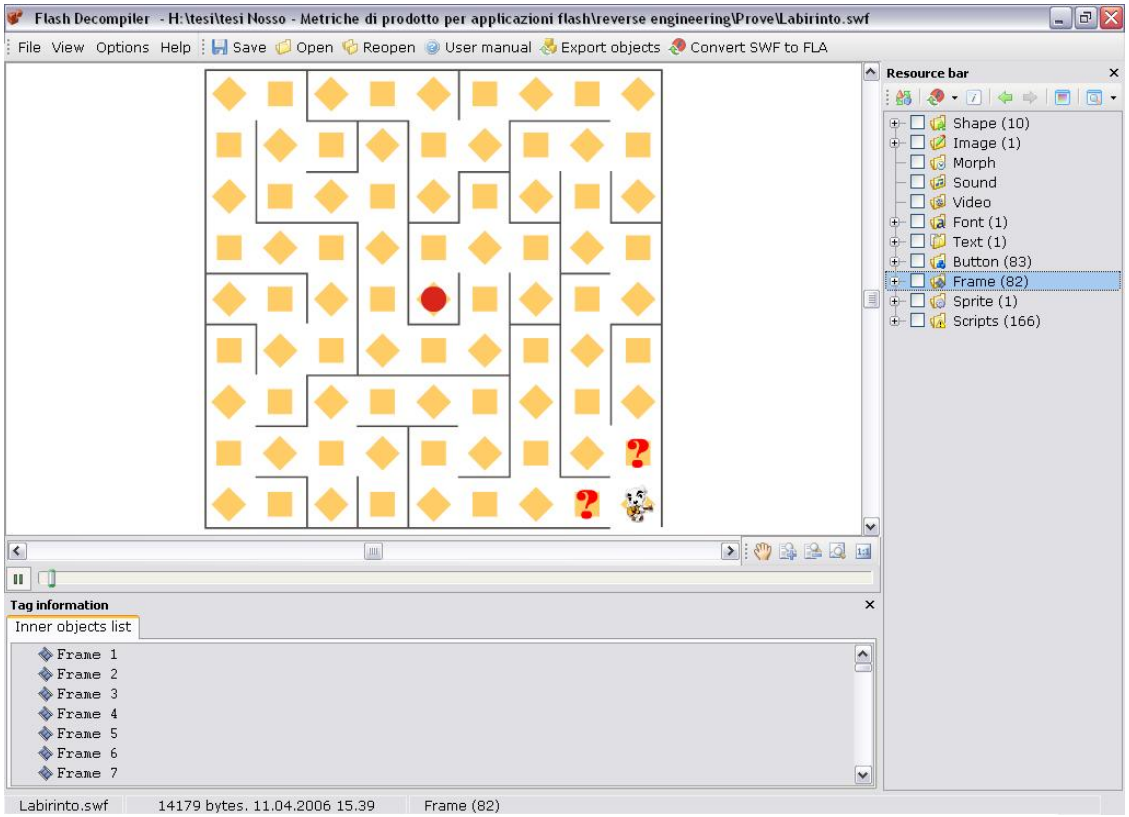

Click di un pulsante che richiede una pagina web

UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II

# Un ulteriore caso di studio (1)

## Estrazione delle informazioni

Applichiamo il Flash Decompiler all'applicazione **labirinto.swf**.




Il decompilatore individua:

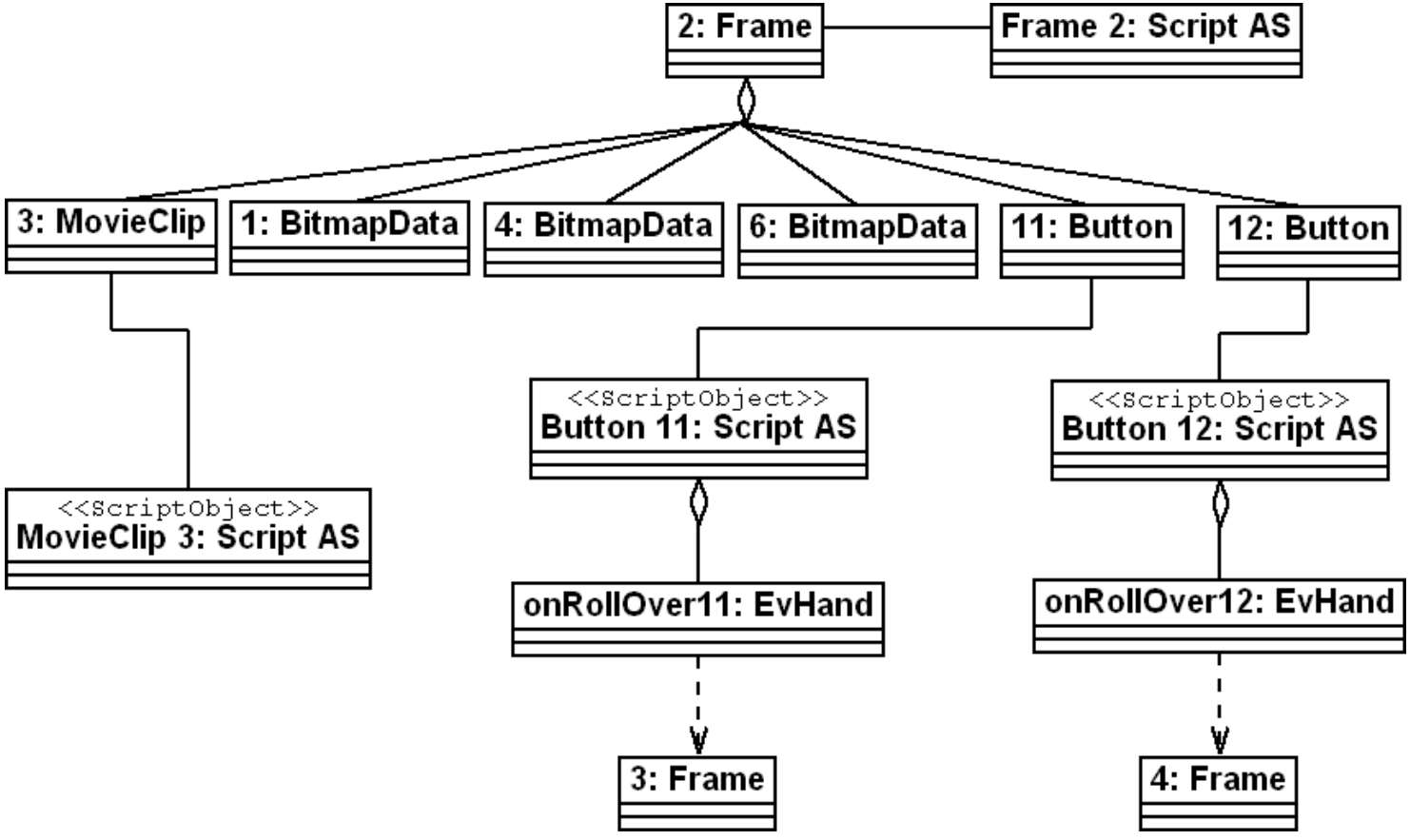
- 10 oggetti di tipo Shape;
- 1 immagine;
- nessun morph, sound o video;
- 1 oggetto di tipo Text;
- 83 Buttons;
- 82 Frames;
- 1 Sprite;
- 166 oggetti di tipo Script.

UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II

# Un ulteriore caso di studio (2)

## Astrazione Object Diagrams

L'animazione esaminata è composta da 82 frames, per ognuno dei quali può essere costruito un Object Diagram di dettaglio, come quello presentato in figura, relativo al Frame 2



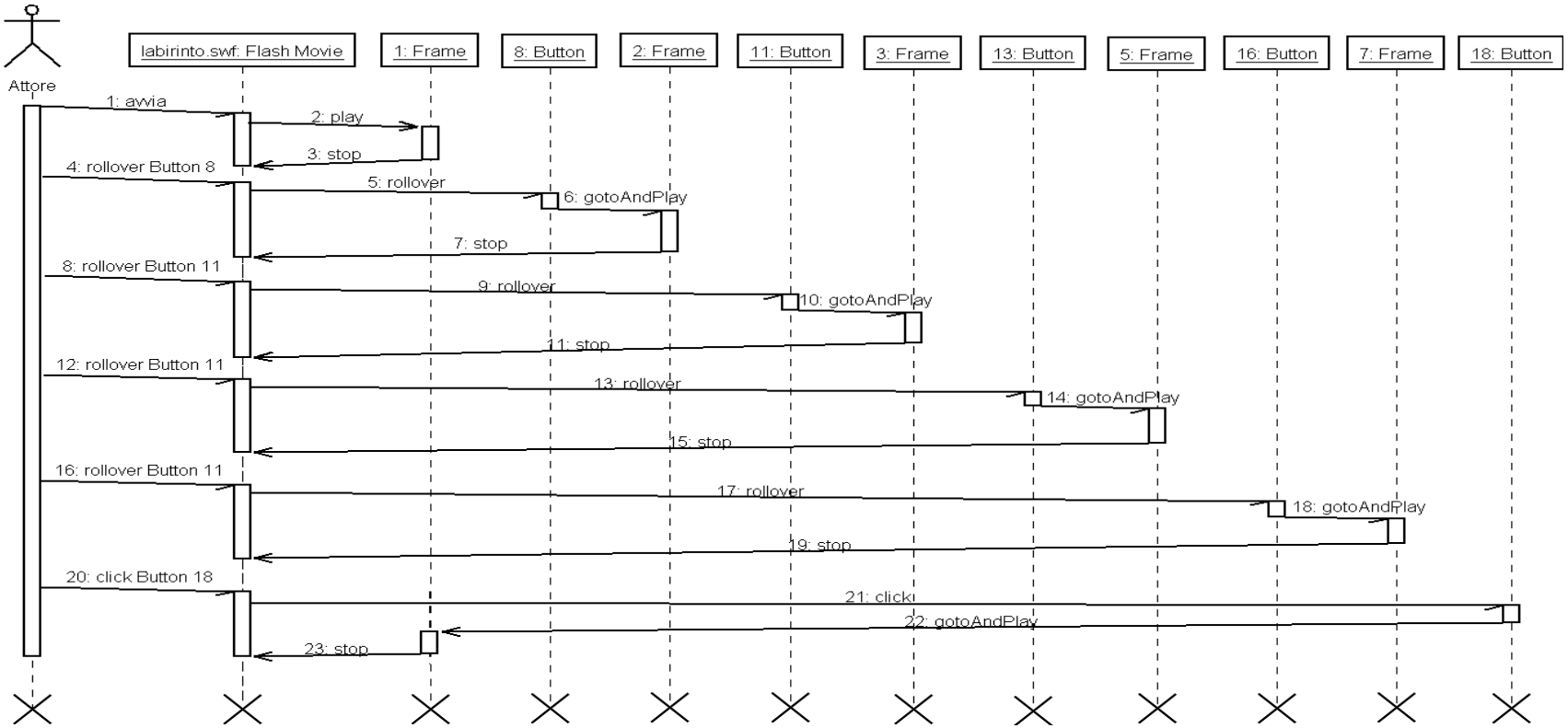
UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II

# Un ulteriore caso di studio (3)

## Astrazione Sequence Diagrams

Per l'applicazione esaminata esistono un numero molto elevato di possibili percorsi nel labirinto. Per ognuno di essi possiamo costruire un Sequence Diagram.

Nel seguito ne presentiamo uno rappresentante una sequenza di mosse che ci porta ad un vicolo cieco.



UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II

# Possibili estensioni future

E’ possibile, infine, prevedere i possibili sviluppi futuri del lavoro illustrato in questa tesi.

- Nella costruzione del meta-modello abbiamo supposto di poter estrarre dall’animazione tutte le informazioni necessarie.

- Alcune le abbiamo ottenute col Flash Decompiler, ma, al fine di poter utilizzare appieno il meta-modello elaborato, sono necessarie ulteriori informazioni non restituite dal decompilatore.

Il passo successivo, quindi, è sicuramente quello di implementare dei tool che possano estrarre informazioni aggiuntive dell’animazio-ne e organizzarle al fine di inserirle, ad esempio, in un database.