The TUNES System Specification

Brian T. Rice, Editor

8th April 2003

Contents

Chapter 1

Overview

1.1 Purpose

This document presents schematic description of the requirements for a TUNES system architecture. This is a working draft, designed to provide precise technical feedback to TUNES project members as design issues are resolved. Implementation strategies are suggested, but not specified.

1.2 Scope

This document specifies the basic requirements required to satisfy the definition of a TUNES system as set out at the TUNES Project website `http://www.tunes.org/`. This is not a tutorial or an implementation guide, although suggestions may be made in order to illustrate and coordinate ideas.

1.3 History

The TUNES Project has existed for many years in an early planning and speculation stage. Several projects exist to advance its goals, but coordination was found to be necessary to help guide the project and solidify the goals.

1.4 Document Conventions

1.4.1 Typography

- Formal terms which have a precise definition within the specification are denoted with a SMALL CAPS style.
- Emphasis is provided through *italics*.

1.4.2 Organization

Since this document is a reference, the order of presentation has not been layed out for introductory purposes.

This document's chapters are separated into levels of detail. Major sections of the system are explained in terms of their core types and operations on those types.

1.5 Conformance

Unlike many standards, which delineate deviation in terms of disallowed behavior within the system under scrutiny, the nature of the TUNES project requires that conformance be tested by verifying that the implementation *allows* everything specified by the standard. Incompleteness is expressed in terms of possibilities that are not supported.

Specifically, if some element of the specification is not explicitly restricted in its scope, then it should be read as stating that there is no restriction of the scope. The simplest example of this is the type of an object within some role in an operation or other type.

1.6 Subsets

This specification defines a standard subset in various sections which is considered suitable for bootstrapping. This subset may be altered to suit dynamic requirements as the initial construction and bootstrap process proceed.

This specification does not define any other subsets, but defines frameworks for publishing subsets.

Chapter 2

The System

2.1 Introduction

A running TUNES system consists of a self-supporting CONFIGURATION of OBJECTs.

2.2 Description

A TUNES system is distinguished from a programming language, an operating system, or various mixtures of these concepts as such. TUNES is first and foremost an *environment*; that is, it is defined by the services it provides rather than the form, structure, or interface that those services take on in a given implementation. The project is moreover an attempt to separate service or interface from implementation in a very broad, systematic sense, while providing a coherent environment in which many implementations may co-exist and provide the same services in different CONTEXTs.

2.3 Requirements

2.3.1 Fully-Reflective

The system must contain at least one standard exectuable EXPRESSION of its full architecture within itself.

2.3.2 Unified

The system must provide standard support for unification of system ABSTRACTIONs. This requires at least that for any two TYPEs of OBJECT within the system and corresponding CONTEXTs, there must exist some standard way to express (or generate the expression of) each OBJECT in the opposite CONTEXT.[1]

[1] This requirement's formulation is flawed.

2.3.3 Verifiable

The system must contain and be able to subject its parts to a means of mechanical verification of assertions within contexts. There must also be a published means of communicating these results when migrating software.

2.3.4 Higher-Order

Any kind of ABSTRACTION is recursively possible over the system and its OBJECTs. That is, OBJECTs created by abstraction operations should be subjectable to further abstractions based on their TYPE.

2.3.5 Self-Extensible

The means of expression must be available within the system to abstract some (*any?*) issue in a system-wide scope.
There must be a sufficient collection/system of ABSTRACTIONs available (but not necessarily imposed) in every area of the system for a meta-system transition in some (*any?*) manner to be possible based on them. Examples include basic syntactic quotation, escaping mechanisms, quotienting in arbitrary directions of abstraction, or some kind of quotient inversion, where a simple domain is carved out of a complex domain by adding some required constraints which they do not all satisfy. This latter example corresponds roughly to a comprehension, of which various sorts exist.
As with any element of these requirements, this kind of capability has to be available in any domain within the system or across multiple systems.

2.3.6 Dynamic

Any operation defined here may be used dynamically without loss of capability. That is, it is expressly permitted for any TUNES system operation to occur in the midst of any computational activity within that system at any time. Any restriction made on what can be done dynamically constitutes the definition of a SUBSET.

2.3.7 Fine-Grained

A TUNES system must place no limit on the ability to eliminate expressive overhead involved in operations on OBJECTs of a specific TYPE, regardless of its kind, within the system.

2.3.8 Fault-Tolerant

A TUNES system must provide some means for assuring that no mismatch or variation of expected behavior will interrupt the system as a whole.

2.3.9 Distributed

Any operation or object should be implementable or re-implementable by a coordination of many other objects without restriction in expressiveness. This applies to physical distribution as well as semantic distribution.

2.3.10 Scalable

Each subsystem defined as a TYPE or an operation must be scalable. That is, for a given CONTEXT, there must exist some facility for scaling down the semantic generality covered by the implementation of that OBJECT to the level of semantics actually employed within the CONTEXT. This is related to the notion of partial evaluation.

Chapter 3

Aspects

3.1 High-Level

3.1.1 Introduction

The High-Level aspect of a TUNES system includes all objects required to define any ABSTRACTION within the system.

3.1.2 Types

Object All elements of the system. Within this document, any unqualified use of this term means any element of a TUNES system. Objects may be manifest or implicit in a particular context, but there will at all times be some mechanism(s) available to make any object manifest within a context.

Attribution An OBJECT relating a *source* object to an *attribute* object via a *key* object. An attribution may be static or dynamically-resolved, and (side-)effecting or not.

Meta-Object The *term* for an OBJECT which deals with some part of the essential characteristics of another. An object can be a meta-object of another object independently of whether or not it provides some function of the implementation of that target object. There are various types of meta-objects. Each meta-object type that deals with a universal kind of aspect of objects may be a meta-object of any object in the system without restriction.

Configuration An OBJECT representing some requirements on the relationships between some collection of objects.

Context Any OBJECT encapsulating the implicit meaning of another object. The nature of a context object determines how an object is expressed within that context. The expression of an object within a context is the object's creation, so there is some definite orientation of the object to that context.
Unlike the term used within human linguistics, there is no natural lack of access

to the original context, so a TUNES system only engages in contextual forensics for external systems or internal, unpredictable systems.

Type Any OBJECT representing a class of properties that another OBJECT or CONFIGURATION of objects satisfies. All objects have types; all objects may have multiple types. The type of an object is in one way self-defining: each object satisfies properties that distinguish it from other objects, and this is precisely what distinguishes objects.

Type System The *term* for a collection of types along with operations on those types and some form of relations between them.

Effect An OBJECT representing a change in the system. These can be provided at various abstraction levels, and the changes themselves may have varying semantics. It is nevertheless the required case that for every effect there must be an expression of it as a function on the system.[1]

Expression An OBJECT representing another in some linguistic CONTEXT.

Side-Effect The *term* for an EFFECT not encoded expressly in some EXPRESSION.

3.1.3 Operations

Apply Couples an ABSTRACTION expression with an input expression.[2]

Abstract (or *lambda*) Creates a function by specifying a REWRITE with an input CONFIGURATION and some body expressions.

Instantiate (or *epsilon*)

Replace

3.2 Meta-Linguistic

3.2.1 Introduction

The Meta-Linguistic aspect of a TUNES system includes all objects which deal with the expression of languages and relation and translation issues between them.

3.2.2 Types

Expression An object representing another in some linguistic context.

Specification An expression describing a configuration.

[1]This is too strong of a requirement, if it implies that all information must be retained when replaced.
[2]Fix this.

3.2.3 Operations

Parse Transform a stream of (character) objects into an OBJECT structure

Expand Code-transformation techniques that are transparent to the semantics of the program specification itself.

Search Control of evaluation order within the limits of the evaluation-order requirements of the semantics of the CONTEXT.

Dispatch Taking each APPLICATION expression, consisting of a function designator and its arguments, and determining according to the environment and its function-definition semantics, and *identifying* the right function-definition or combinations thereof to APPLY.

Rewrite Applying EFFECTs specified by the program to the system.

Respond Communicating the effects of some REWRITE to the calling context.

Translate

3.3 Interface

3.3.1 Introduction

The Interface aspect of the TUNES system includes all objects whose purpose relates to the display of information for a user and meaningful interaction with users.

3.3.2 Types

User An OBJECT representing a particular human user or agent thereof that interacts through a TERMINAL device. User objects have an associated environment which carries the vocabulary and preferences specific to that user.

Terminal An interaction component of the system, consisting of at least one input and one output device.

Gesture A unit OBJECT of input or output interaction. The TERMINAL device's characteristics determine the possible granularity of these objects.

Command An OBJECT representing a single event or combination of events which have been given some meaning by their presence within some interaction CONTEXT.

Activity An OBJECT representing a user's model of some sequence of events directed towards a perceived specified objective.

Medium An OBJECT representing a particular interaction device, with its characteristics, behavior, and state. All objects of this TYPE are subject to a generic protocol which allows for abstraction within the limits of the device capabilities.

Display An OBJECT representing an abstract rendering device.[3]

Color An OBJECT representing a visual color, a kind of DESIGN.

Style An OBJECT representing a set of options and settings for rendering something on a DISPLAY.

Region An OBJECT representing a geometric space of some dimension.

Design An OBJECT representing any graphical pattern. Designs may be composed with each other into final designs.

Presentation An OBJECT which contains presentation aspects of the rendering of some object. Presentation objects may have formatting options and their own user-centric type notion separate from the type of the presented object.

Morph An object whose context is a display. Objects of this type are those which offer the user some form of direct manipulation and do not inherently concern the presentation of some other object.

World An object representing some site on a DISPLAY.

Portal An object representing some channel between sites or a bus among sites on a display.

3.3.3 Operations

Draw Rendering a display OBJECT onto a DISPLAY with some particular STYLE.

Print Rendering an OBJECT onto a DISPLAY in a form suitable for direct input to some reader.

Read Creating an OBJECT from some input description.

Present

Request

Accept

3.4 Migration

3.4.1 Introduction

The Migration aspect of a TUNES system includes all objects which describe the identity and relate to mechanisms for moving or duplicating objects between any kind of contexts.

[3]Is this redundant with a MEDIUM?

3.4.2 Types

Module A CONFIGURATION of OBJECTs with formal requirements for comprehension and formal provisions.

Site A source or target of communication.

Protocol A medium of communication. Precisely, a language/encoding.

Stream A possibly endless sequence of objects, sent between sites.

Packet A single element of a stream.

Memory Region A subset of a storage device with a particular management type for its memory.

Space [4]A sharable OBJECT which provides a means of accessing publications.

Reference An encoding of a remote OBJECT; generally, a program to retrieve an object for *use*.

Persistent Store

Version

3.4.3 Operations

Reify Takes some SPECIFICATION of the target CONTEXT, and removes an OBJECT from a source CONTEXT based on the required elements. It *slices* the OBJECT from the CONTEXT, and represents the relevant parts to send to the target CONTEXT.

Communicate Renders the MODULE in some medium that both the source and target CONTEXTs advertise that they support, and actually transports the MODULE's contents as necessary.

Absorb Takes a description of some MODULE to receive, and evaluates as necessary to *install* the objects in the target CONTEXT.

Publish Registers some MODULE within a SPACE, with a TYPE and a VERSION known to some shared server SITE which can mediate the exchange. SPACEs themselves can notably be published.

3.5 Low-Level

3.5.1 Introduction

The Low-Level aspect of a TUNES system involves all objects whose sole role involves the implementation of other objects in the system.

[4]Revise this term's name.

3.5.2 Types

Bit A single discrete unit of memory with two possible states.

Word A vector of BITs defined as a unit of memory for some (possibly abstract) machine.

Memory A quantity of information managed and storable by some device.

Processor An operational machine that processes information stored in MEMORY.

Channel A structure through which information is passed between two SITEs or OBJECTs, which can have state and modes. Channels can be concrete or abstract, and abstract channels can be emulated by other channels through agreed protocols.

Bus A shared communications medium, having many publishers and subscribers to information. A bus can often be considered in terms of many cooperating CHANNELs.

Thread An OBJECT representing a unit of execution, possibly with an associated voucher for PROCESSOR resources.

Encoding An OBJECT representing a particular TYPE of low-level representation for some other type of OBJECT.

3.5.3 Operations

Bootload

Call

Allocate

De-allocate

Encode

Decode

Chapter 4

Elements

4.1 High-Level

4.1.1 Objects

4.1.2 Attributions

4.1.3 Effects

4.1.4 Abstractions

4.1.5 Applications

4.2 Meta-Linguistic

4.2.1 Grammars

4.2.2 Evaluators

4.2.3 Dispatchers

4.2.4 Responders

4.2.5 Transformers

4.3 Interface

4.3.1 Terminals

4.4 Migration

4.5 Low-Level

Chapter 5

Subsets

5.1 Definition

A TUNES system SUBSET is a specified restricted group of capabilities that fulfill the common criteria of functionality of some TUNES aspect or subsystem. The intended purposes of a subset may be to bootstrap or connect from legacy systems; both purposes can be satisfied simultaneously. Another purpose of a TUNES subset may be to enforce some invariants for the sake of providing an environment customized to a particular idiom or style, and easing the implementation's load in rendering the environment efficiently.

5.2 Core/Bootstrap Language Semantics

5.2.1 Introduction

The "HLL-" language is a defined SUBSET of the TUNES system High-Level aspect. The primary design considerations include proper allowance for extension into the full system, avoidance of limiting semantics or features, and easy of implementation.

5.2.2 Semantics

5.2.3 Syntax

5.3 T3P/TM3P Module Encoding Protocol

5.3.1 Introduction

The TUNES Package Publishing Protocol (or "T3P") is a defined initial ENCODING for the Migration aspect of the TUNES system. The primary design considerations include independence of the communications medium, universal extensibility at least to the level of the HLL-, and generality in negociating information transfers.

5.4 Initial Standard Publication System

5.4.1 Introduction

Index

Bibliography

[1] The TUNES Project Website `http://www.tunes.org`. François-René Rideau.

[2] The Common Lisp Hyperspec, derived from the ANSI Common Lisp standard (X3.226-1994), at `http://www.xanalys.com/software_tools/reference/HyperSpec/`. Common Lisp committee J13; Project Editor, Kent M. Pitman. 1994.