

USENIX Association

Proceedings of the XFree86 Technical Conference

Oakland, California, USA
November 8–9, 2001



© 2001 by The USENIX Association
Phone: 1 510 528 8649

All Rights Reserved
FAX: 1 510 548 5738

For more information about the USENIX Association:
Email: office@usenix.org

WWW: <http://www.usenix.org>

Rights to individual papers remain with the author or the author's employer.

Permission is granted for noncommercial reproduction of the work for educational or research purposes.
This copyright notice must be included in the reproduced paper. USENIX acknowledges all trademarks herein.

Salamander: The Quest to Build a Useful Handheld Computing Environment

Alexander Guy
Andern Research Labs
a7r@andern.org

Abstract

As of late, the availability of commercial, off the shelf, handheld devices has grown dramatically. It is now possible to walk into most consumer electronic stores, and purchase a handheld device capable of running a fully fledged operating system such as Linux or NetBSD.

The only major differences found in these handheld devices, as compared to their desktop counterparts, are related to hardware. The computing and storage resources available to users and applications are under much different constraints. In addition, input and output mechanisms are generally quite different. Because of this, although the existing application base is available (e.g. GNOME, KDE, and other X applications), their use is in many cases infeasible (although much code and experience can be gained).

Salamander is a project aimed at developing a useful handheld computing environment. Based on UNIX-like operating systems, with X as the windowing environment, Salamander exists to provide an open-source suite of Personal Information Management (PIM) tools and a support structure for applications on current and next-generation handhelds.

1 Introduction

Handheld devices have advanced over the years. From the Atari Portfolio, through to the Palm Pilot, and Apple Newton, handheld hardware has progressed rapidly. The latest devices to emerge, offer numerous new possibilities. HP's Jornada and

Compaq's iPAQ series of handhelds represent the current state-of-the-art for consumer handhelds.

Open-source development on handhelds has been growing by leaps and bounds. Thanks to support from Compaq, and the community that's been generated around the handhelds.org website, the iPAQ line of handhelds has become a formidable development platform for Linux-based handheld applications.

Presently, Linux on the iPAQ supports all APIs provided on its desktop counterparts. Because of this, most applications may be ported to these handheld platforms with little or no modification. A full implementation of X is available, as well as such classics as GNU's libc and Python. Several Linux distributions have been developed for, or ported to, the iPAQ, making conversion of existing WinCE-based iPAQs to Linux quite easy[hhw].

While there are numerous applications and user environments available for X (such as GNOME and KDE), most provide less than optimal solutions given the environment they're being executed in. Most of the existing modern code-base was designed for use without the constraints that a handheld device provides. Aspects such as persistent storage, memory, and screen real estate are at a much higher premium than found in a typical UNIX workstation¹. In addition, many existing applications weren't designed for use with the forms of input provided by handhelds (e.g. "one-button" touchscreen, tiny keyboard, voice recognition).

Salamander attempts to provide a suite of handheld applications that cover the common functionality

¹Thanks to X's beginnings, there have already been numerous test cases that show X to be a viable windowing solution for resource-poor devices.

provided by existing handheld devices and software, while taking advantage of the new possibilities offered by being based in a UNIX-like environment.

1.1 Goals

The basic requirements for Salamander consist of the following:

- **Usefulness.** Many of today's "productivity" devices seem to only cause additional, undue, complexity. Users must benefit from Salamander's involvement in their affairs.
- **Reliability.** This is a priority, as Salamander must provide a stable environment that users can depend on for storing and retrieving vital information.
- **Simplicity.** Providing a set of easy to understand applications that can easily interact with each other is far better than the creation of monolithic, independent, applications.
- **Interoperability.** Salamander must easily and completely, interact and interoperate with existing available devices, such as cellular phones and other PDAs.
- **Ease of Development.** Creating additional applications that can homogeneously interact with the existing Salamander environment is a must.

2 Implementation

Deciding which tools to use for prototyping took a bit of thought. Our criteria for choosing code and tools to work with came down to the following:

- Allowed For Rapid Prototyping
- Mature Code-Base
- Low Resource Utilization

When working on developing a set of new technologies, it is a must that they get to a state of evaluation as quickly as possible. Too much time spent on debugging, or on rewriting existing code that doesn't quite fit, results in less time for actually focusing on the project goals.

While developing a system from scratch is tempting, for numerous reasons there are severe drawbacks to having too many pieces of foundation in flux. For Salamander's development, we wanted to use as much existing code as possible; only performing original development, or serious modification, when no appropriate piece of software already existed.

Because of the hardware environment that Salamander is designed to run in, resource utilization is a serious concern. Elements used must have: reasonable memory usage, low CPU requirements, and a small non-volatile storage cost.

2.1 Prototyping Choices

X was selected very early on as the best choice for the windowing environment. Not only is the codebase mature and well understood, it is also under active development. Recent efforts by Keith Packard [Pac01] and Jim Gettys [GP01] have done quite a bit to enhance X's usefulness in a PDA environment.

Python v2 was chosen as the environment in which to develop most of Salamander's original code. Its quick development time, plethora of existing modules, and existence in a ready to use and complete form on handhelds were key in making the choice to use it. Python also provides an easy mechanism for developing bindings to C libraries, making integration with existing code a snap. In addition, the developers starting the project, all had previous development experience with Python, making the learning curve non-existent.

Although not the ideal solution for handhelds, GTK+ was chosen as the widget set to use for interacting with X. This decision was made primarily due to PyGtk, James Henstridge's excellent GTK+ bindings for Python. GTK+ has existed for quite a while and has had numerous projects developed against it.

Sleepycat's BerkeleyDB libraries were chosen as general tools for data storage. Although XML provides excellent interoperability traits, it leaves something to be desired as far as storage space efficiency and CPU usage for parsing goes. BerkeleyDB provides an excellent binary storage format, with a well understood and commonly available API for interaction.

3 Architecture Philosophy

Salamander follows the UNIX approach of building simple and robust pieces that can be connected together to perform complex work. Each one of these pieces, called a "component", is a peer within a community of components, that interact with each other to actually perform work.

4 Core Components

Salamander's core components represent the baseline functionality upon which all other components in the system depend. Core components are guaranteed to be available in any Salamander installation and provide the basis for all other functionality.

4.1 Dispatch

The passing of messages makes up the heart of the interaction between different Salamander components. Messages are routed in a switched manner, through a central switchboard mechanism called Dispatch.

Dispatch is responsible for handling all communication between connected components in the system, called "peers". Each peer is assigned a unique identifier upon connection and may, at the peer's discretion, opt to have other identifiers assigned, that may or may not be shared by other peers (similar to multi-cast, or channel-based concepts).

Messaging is important for Salamander because of the philosophy behind components. Instead of having loadable embedded objects, each component acts as a separate entity, capable of being requested to do a specific task. When one component requires something of another, a dialogue is started between said components, and the required steps are taken.

Access control is handled by the initial connection to Dispatch. Once a peer has authenticated itself to Dispatch (currently handled using permissions on UNIX-domain sockets), its messages are considered valid, and trustworthy.

4.2 Persistent Object Store

The Persistent Object Store (POS) represents one of the core components of Salamander. Using the POS, components are given a mechanism to not only persistently store information (in the form of discreet objects, that can be referenced, and linked together), they're also given a simple way to share large groups of objects between each other.

Objects within the POS have a given class, which defines what members objects are permitted to have (i.e. schema is enforced). There are many existing cases of database and interfaces that provide this type of functionality, but using them in the environment presented proved to be too much of a difficulty.

Communication with the POS is handled just like any other intercomponent communicate, through the message passing system. Classes are defined, objects instantiated and destroyed, and members modified, all through messages passed from components.

One of the most interesting aspects of the POS is the ability to link objects. Conceptually, the objects in the POS interconnect to make a large unbalanced object-graph, with different nodes interconnecting depending on the schema that each follow. This provides an interesting concept for handheld devices, because it means that information association can easily be stored and taken advantage of.

Take the example of a scheduled appointment. Given the ability to link to different objects within the POS, a given appointment could interconnect with the objects representing the people being met, as well as the location for the meeting.

This ability to link POS objects together is one of Salamander's main strong points. It allows for lowered redundancy, both as far as computing resources, and as time required for users to perform work.

5 Applications

A user environment is of little use with only infrastructure developed. The initial creation of the PIM tool suite serves as the litmus test for Salamander's base framework. Applications developed for the most part have been analogous to software found in other environments, while taking advantage of some of the new functionality afforded by Salamander's manner of implementation and general philosophy.

5.1 Mingle

Mingle is Salamander's answer to the typical PDA contact manager. Using the POS, Mingle provides a mechanism for creating, deleting, and editing contact entries. Due to the model of the POS, Mingle is provided with some powerful linking capabilities.

Contact entries can be linked with JPEG images stored in the POS, allowing faces and pictures to be associated with a given person. Multiple phone-number and physical address objects can be linked from a contact, and shared between contacts (interesting for storing directions to a given location, or keeping notes on phone calls made to a specific phone number).

5.2 Task

Task exists to allow the creation and prioritization of required activities and projects. Tasks may remain unconnected, or associated with an instant or period of time.

Tasks may also be used to track time expended performing a given operation, allowing users to track "where the time went".

Task depends on, and is tightly coupled with, Chronicle for all the time-keeping related activities.

5.3 Chronicle

Chronicle is responsible for providing the user with an interface to add, organize, and/or remove, time-based events and objects associated with a given moment in time.

Information about the current state of the user's schedule is gleaned from any pertinent objects

stored in the POS. In a day view, Chronicle would display the current tasks related to the current day, any scheduled events (along with their durations), and any tasks that must take place at a certain time.

6 Current and Future Status

Salamander is currently undergoing rapid development. New functionality is being added daily, to extend the usefulness of the environment. While still in the early stages of development, results are promising. For more information on the state of the project, please see <http://www.andern.org/salamander/>.

Acknowledgements

Thanks to Compaq for supporting the handhelds.org website, and putting such exceptional resources behind the development of Linux on the iPAQ. Thanks to all the existing open-source projects that have made the creation of Salamander a possibility. I'd also like to thank Gareth Greenaway, Bert Vermeulen, and Ian Hall-Beyer for reviewing this document.

References

- [GP01] Jim Gettys and Keith Packard. The X Resize and Rotate Extension - RandR. In *FREENIX Track, 2001 Usenix Annual Technical Conference*, Boston, MA, June 2001. USENIX.
- [hhw] Handhelds.org. <http://www.handhelds.org/>.
- [Pac01] Keith Packard. Design and Implementation of the X Rendering Extension. In *FREENIX Track, 2001 Usenix Annual Technical Conference*, Boston, MA, June 2001. USENIX.