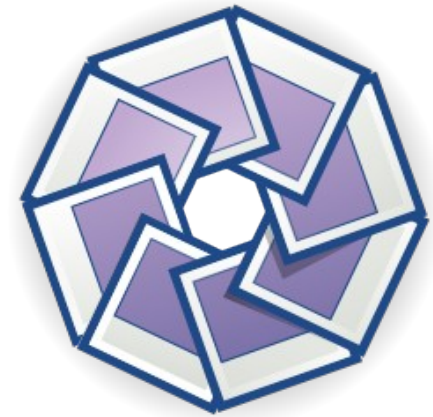


Andrzej Wytyczak-Partyka
email: andrzej@wytyczak.com
jabber: andrzej@wytyczak.com



F-Spot face tagging feature

a Google Summer of Code 2007 project

Table of Contents

1. Project description.....	3
1.1 UML use case description.....	3
2. Benefits for the Gnome community.....	5
3. State of the art.....	6
3.1 Face detection methods.....	6
5. Deliverables and timeline.....	8

5. Why me?.....	12
Indexes.....	13
Bibliography.....	13
Illustration Index.....	13

1. Project description

Vision.

For some time now we are observing a tendency for the desktops to become more and more intelligent. The Gnome Desktop (further: Desktop) is no different, the integration between Gaim, Evolution, Beagle, and others, is a great example of that. The Desktop is aware of user's contacts, it keeps track of conversations, documents, etc. But that's old news – here and now is Voice over IP, video calls, digital media sharing and Web 2.0. These technologies are not yet integrated with the Desktop – how could it „understand” voice conversations, or video clips uploaded to our Google Video account?

The idea behind this project is to make the Desktop, and F-Spot in particular, „understand” the data it deals with. My vision is to provide means for full integration between F-Spot, Beagle, Evolution Contacts, Gaim, and do that by enabling F-Spot to recognize faces of our friends in the pictures it collects in it's albums. F-Spot has to recognize the faces and correlate them with people it knows. If an unknown face is encountered – the user has to be prompted for input, to tell F-Spot whose face it is. This way, by remembering whose faces are on each picture, F-Spot would generate an index, that could be queried by F-Spot itself and other programs, like Beagle.

This project.

Within the Summer of Code I would only implement the detection and tagging of faces – the rest would be left for later work.

1.1 UML use case description

There are many use cases within the F-Spot, but really only one concerns this project – the „Tagging photos” use case. I will detail it below. It can be called in several ways, during the import process, while browsing the album, while browsing the individual pictures. I will only describe the scenario when this use case is called while browsing a single picture.

1. Tagging photos

The user tags photos, the tags can be of various kinds – they can denote the place in the picture, the time, or the people on the picture. After the user enters the tags they should appear in the database.

General scenario :

2.1 User enters tags for the picture

Extensions :

Ad 2.1 If the system has detected any faces on the picture, bounding boxes appear (see III. 2) and if the faces have been identified also the appropriate tags appear automatically over the bounding boxes. If the system doesn't recognize a face it's denoted by a '?' sign over the bounding box.

Ad 2.1 If the user is not happy with the system's suggestions it is possible to correct the bounding boxes' locations and/or add/delete boxes.

Outcome :

The photo's tags are updated. The face database is updated. The faces are associated with their respective tags.

2. Benefits for the Gnome community

The Gnome community would receive an impressive feature, unseen (as far as I know) in any other picture management software. The Desktop would simply become a smarter and more neat place :)

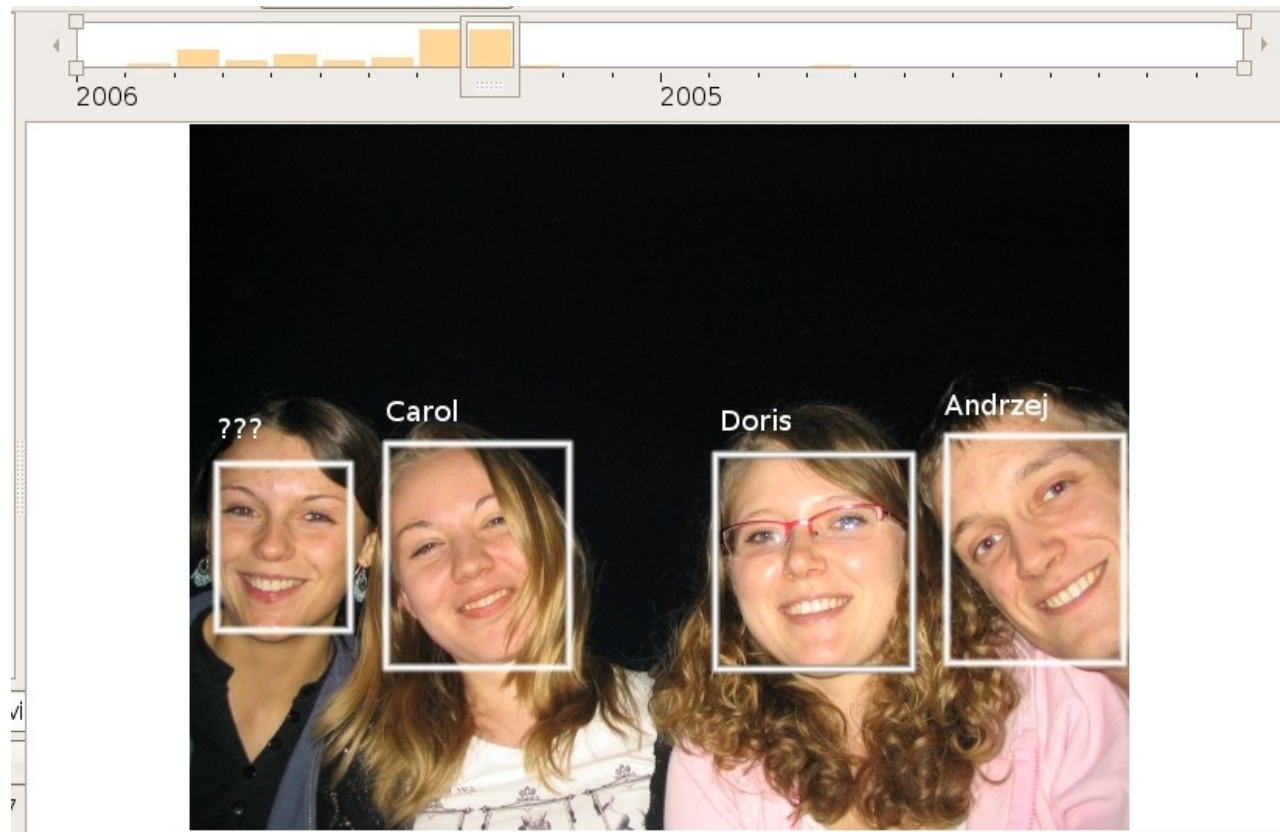


Illustration 1: Effect of face detection (bounding box) and face recognition (name above the box), mockup

3. State of the art

The buzz-word for this project is face recognition, although it's not just it. To recognize a face, it has to be found on the image first, and that is face detection. I will briefly describe the current face detection methods and present a certain method that I believe would be good for this project.

3.1 Face detection methods

The face detection algorithm's goal is to select image regions that possibly contain faces. Its work is represented as the bounding box over a certain region of the analysed image (Fig. 1). The coordinates of those regions are later sent to the face recognition algorithm. There are several known methods for face detection, they can be generally classified into three groups – color-driven, feature-driven and motion-driven. The latter we won't consider, as we will be dealing with static images only.

- By color

This method alone is more like a general skin detector than a face detector. It allows fast segmentation of regions containing skin, based on simple classification. The set of skin colors has to be determined before the algorithm is usable. Usually the HSV / YIQ / YCbCr color models are used because the RGB model is lighting dependent and the classification would give different results depending on the lighting conditions. In the chrominance - luminance models the luminance can be omitted and the algorithm deals only with the chrominance signals.

- By features

Algorithms based on feature detection are in general making use of aspects like the shape of the face (approximately oval) or location of eyes, nose, mouth, chin and their relations.

For the purpose of the considered system some general reservations as to the algorithms have to be made.

- Speed

The algorithm should perform it's duties in a timely manner. Realtime would be a plus.

- Accuracy The algorithm's accuracy should be satisfying for the user. A 70% recognition rate would be a minimum.

Because of the negative correlation between speed and accuracy the user should be able to determine which one of these factors is more important.

For the detection I have selected the algorithm (feature-based) presented by Viola and Jones [3], it performs in realtime with a speed of 15 frames per second. The authors also claim a 1 per 14084 false positive with a 95% detection rate, so both reservations (for speed and accuracy) are met by this algorithm.

Intel's Open Computer Vision Library has a face detection algorithm [4] based on Viola and Jones's work.

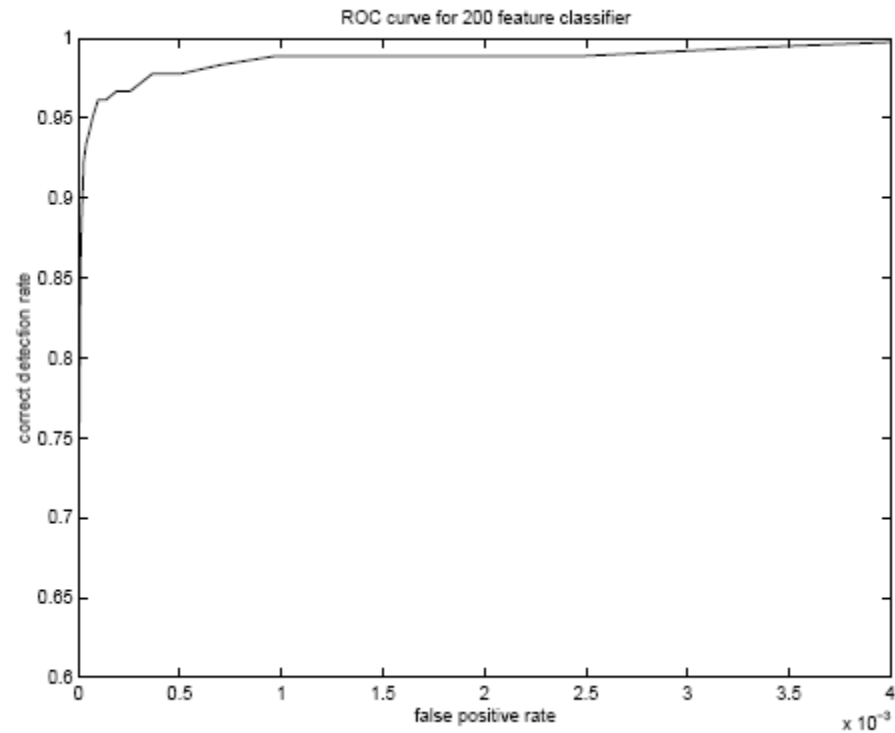


Illustration 2: Viola and Jones' detection algorithm characteristics

5. Deliverables and timeline

There are several modules within this project, It is especially important to discuss the UI with the community (i.e. prepare some mockups and discuss them), so I have planned to work out the guidelines for this module during the Interim period („Work out the UI guidelines”). Secondly there is the face-detection algorithm itself, it's implementation will be done within the „Face detection algorithm” part of the code period. The third, and the last part, is the glue between the UI & the face detection algorithm which is more or less related to the picture tagging procedure and is represented as the „Hook-up” in the timeline.

I have also setup some milestones for the project, like „UI design ready”, „Face detection beta”, „UI beta”, „UI final”, „Hook-up beta” and „Hook-up final”. Each of these milestones is supposed to be a point where the current status of the project is discussed with the community and time is given between beta and final milestones to make improvements.

The final period is the extra time I give myself in case any part of the project slipped of the schedule, it is also the time for a final discussion and improvements.

	Name	Start	Finish
1	Interim period	April 9	May 28
1.1	Work with the community	April 9	May 23
1.2	Dev env setup	April 12	April 18
1.3	Code reading	April 18	May 3
1.4	Work-out the UI guidelines	April 20	May 19
1.5	<i>UI design ready milestone</i>	<i>May 21</i>	<i>May 21</i>
2	Code period	May 22	August 7
2.1	Face detection algorithm	May 29	July 6
2.2	<i>Face detection beta milestone</i>	<i>June 29</i>	<i>June 29</i>

2.3	Mid-term code upload	July 9	July 9
2.4	UI	July 2	July 20
2.5	<i>UI beta milestone</i>	<i>July 13</i>	<i>July 13</i>
2.6	<i>UI final milestone</i>	<i>July 20</i>	<i>July 20</i>
2.7	UI & algorithm hook-up	July 21	August 4
2.8	<i>Hook-up beta milestone</i>	<i>August 3</i>	<i>August 3</i>
2.9	<i>Hook-up final milestone</i>	<i>August 7</i>	<i>August 7</i>
3	Final period	August 8	August 20

The schedule is shown below as a Gantt chart.

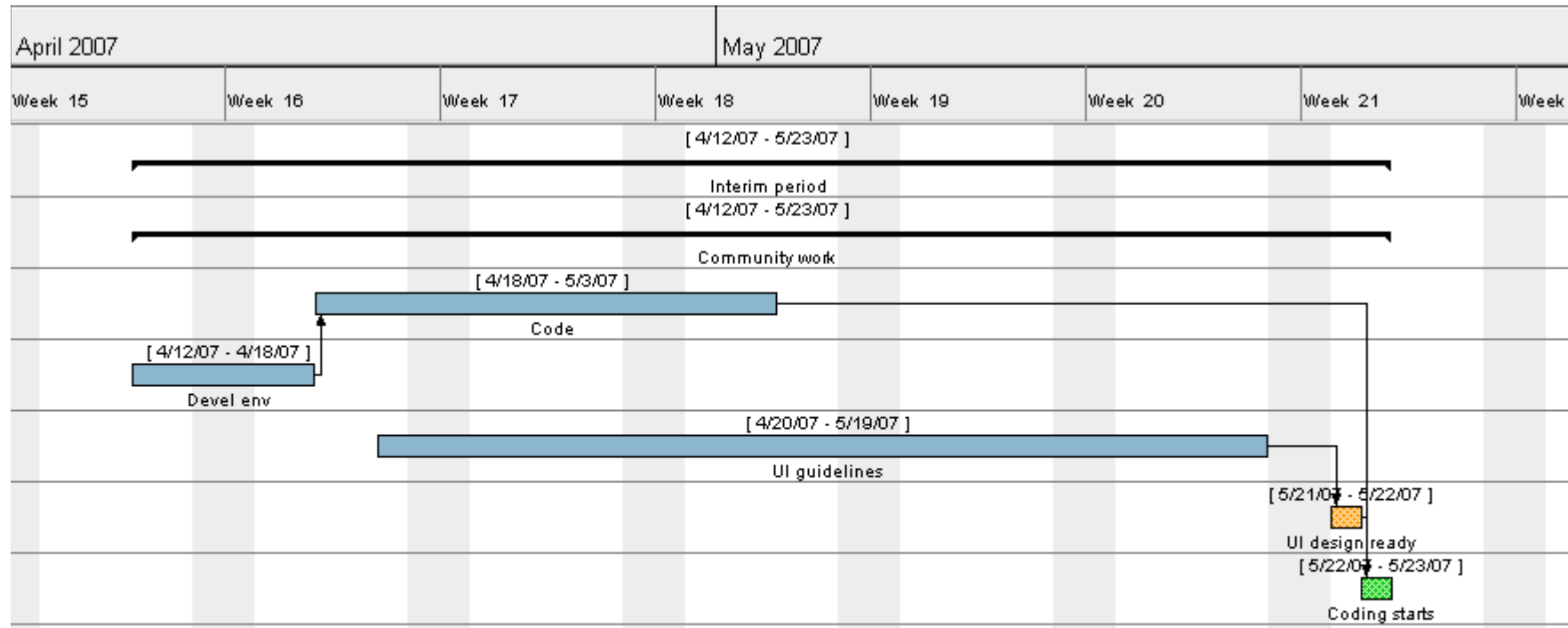


Illustration 3: Kick-off phase : April 9th - May 28th, Gantt chart

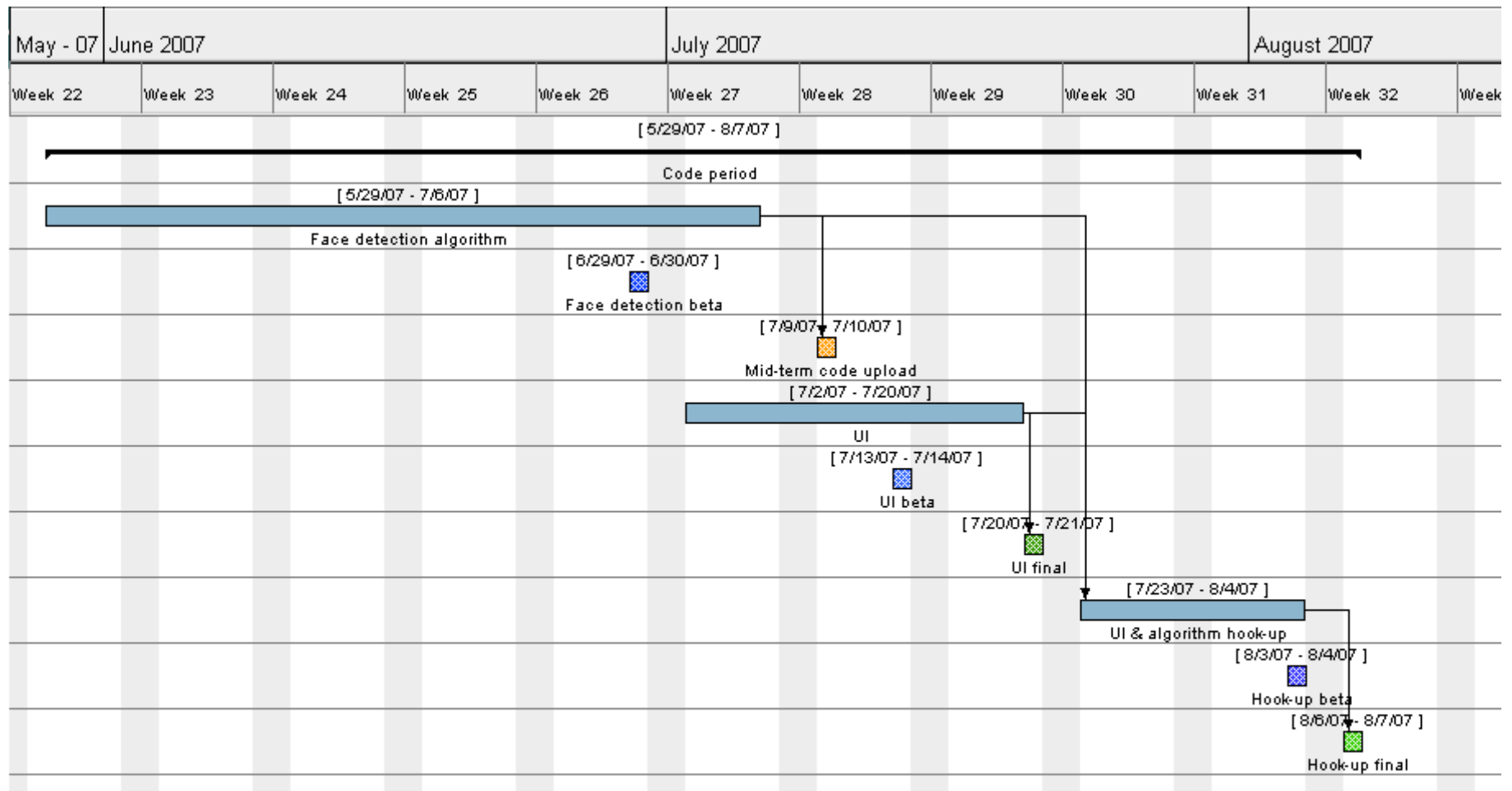


Illustration 4: Code phase Gantt chart

5. Why me?

I have a vision. I strongly believe this project can make the Desktop a better place. If it succeeds it should revolutionize the way album software works. Some experience in the area of face recognition gives me a good start - in my past work I have written a face recognition algorithm, based on [1] and proven it [2] to perform well in terms of computational complexity. I have chosen the Eigenfaces method for face recognition, because of it's simplicity combined with high recognition rates (over 70%). I intend to present some of the results of this project in my thesis, which gives me extra motivation.

I'm a 4th year computer science student at Wroclaw University of Technology, Faculty of Electronics and in my 2nd year of medicine at Wroclaw Medical University. I have good knowledge and some experience in programming, proven among others, during the 2005 Summer of Code when I coded the Tab Browsing feature for OpenOffice.org. During my university course of computer science I have received highest marks in programming and image processing related courses. During the last couple of year's I've been also working as an IT contractor / consultant in various local companies. I have experience in project management and software design. In my spare time I'm writing fun code (like my Evolution libnotify patch) and experimenting with hardware.



Illustration 5: The Evolution libnotify patch at work

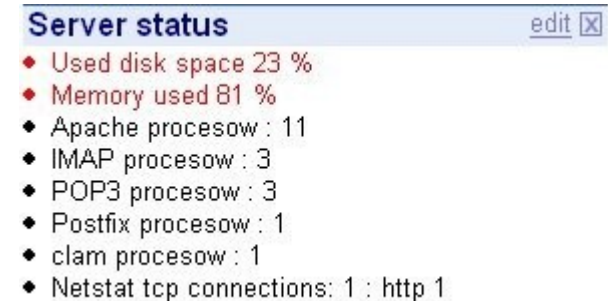


Illustration 6: Google homepage server status gadget

Indexes

Bibliography

- 1: Turk Pentland, Eigenfaces for recognition, 1994
- 2: Baranski Walkowiak Wytyczak-Partyka, Computational complexity reduction in face recognition, 2007
- 3: Paul Viola and Michael Jones, Robust real-time object detection, 2002
- 4: Intel, Learning-Based Computer Vision with Intel's Open Source Computer Vision Library, 2005,
http://www.intel.com/technology/itj/2005/volume09issue02/art03_learning_vision/p04_face_detection.htm

Illustration Index

Effect of face detection (bounding box) and face recognition (name above the box), mockup.....	5
Viola and Jones' detection algorithm characteristics.....	7
Kick-off phase : April 9th - May 28th, Gantt chart.....	10
Code phase Gantt chart.....	11
The Evolution libnotify patch at work.....	12
Google homepage server status gadget.....	12