# Handbok Rocs

**Tomaz Canabrava**
**Andreas Cord-Landwehr**
**Översättare: Stefan Asserhäll**

# Innehåll

**Sammanfattning**

Rocs är ett grafteoriverktyg.

# Kapitel 1

# Inledning

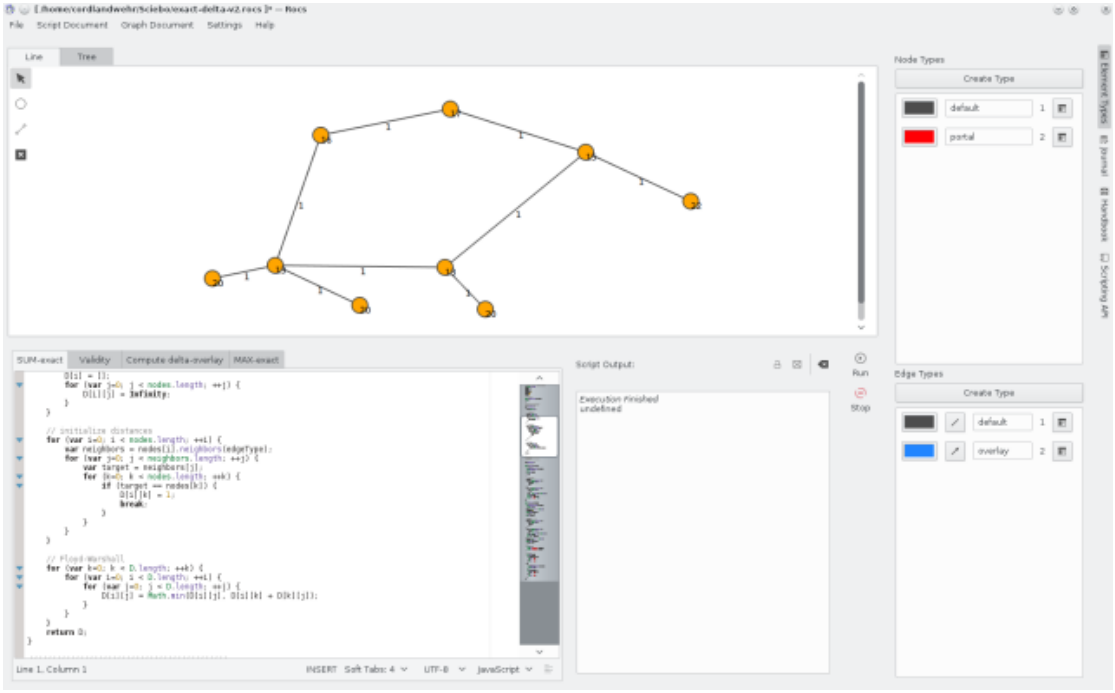
Det här kapitlet ger en översikt av de centrala funktionerna och typiska arbetssätten. De viktigaste delarna är Avsnitt 1.2 och använda kapitel 3, som tillsammans bör göra det möjligt för alla nya användare att börja arbeta med Rocs.

## 1.1 Syfte, målgrupp och arbetsflöden

Rocs är ett verktyg för grafteori avsett för alla som är intresserade av att konstruera och analysera grafalgoritmer. Mer specifikt är de:

- föreläsare som vill demonstrera algoritmer för sina studenter,
- studenter och forskare som vill se hur deras algoritmer fungerar, och
- alla andra som är intresserade av datastrukturer och algoritmer.

För alla tillhandahåller Rocs en lättanvänd grafisk editor för att skapa grafer, ett kraftfullt skriptgränssnitt för att beräkna algoritmer, och flera olika hjälpverktyg för simulering, experiment, och export av grafer. Det typiska sättet att använda Rocs är att skapa en graf, antingen för hand (dvs. dra noder och bågar till skrivtavlan) eller genom att använda en av grafgeneratorerna. Grafalgoritmer kan implementeras och beräknas för den skapade grafen och alla ändringar som algoritmen utför syns omedelbart i grafeditorn.



## 1.2 Rocs i ett nötskal

Varje session i Rocs är ett projekt: när Rocs startas skapas ett tomt projekt, och när ett projekt läses in blir det till det aktuella projektet. Härigenom består själva projektet av *grafdokument*, *skript eller algoritmer* och en *journal*.

### 1.2.1 Grafdokument

Ett grafdokument representerar en skrivtavlas innehåll i grafeditorn. Det innehåller information om de användardefinierade nod- och bågtyperna, deras egenskaper och de redan skapade noderna och bågarna. Det vill säga, Rocs förstår mängden av alla noder och bågar i ett grafdokument som formar en graf (inte nödvändigtvis förbunden). Allting som hör till ett grafdokument är tillgängligt i skriptgränssnittet via det globala objektet **`Document`**.

### 1.2.2 Bågtyper

I vissa scenarier består grafer av olika bågtyper (t.ex. en oriktad graf plus trädbågar beräknade av en bredd först-sökalgoritm), som ska hanteras och visas på olika sätt. I detta syfte går det att definiera godtyckliga andra bågtyper, förutom standardtypen. Varje bågtyp har sin individuella visuella representation, dynamiska egenskaper och kan ställas in att antingen vara oriktad eller riktad. Skriptgränssnittet tillhandahåller bekvämlighetsmetoder för att specifikt komma åt bågar av specifika typer.

### 1.2.3 Nodtyper

I likhet med bågtyper, går det att definiera olika nodtyper i en graf (t.ex. för att ge vissa noder särskilda roller). Varje nodtyp har sin egen visuella representation och dynamiska egenskaper.

### 1.2.4 Egenskaper

Varje nod- eller bågelement kan ha egenskaper. Egenskaperna måste ställas in för motsvarande nod- eller bågtyp. Egenskaper kan identifieras och kommas åt med sina namn, och kan innehålla godtyckliga värden. För att skapa nya eller ändra befintliga egenskaper, använd sidoraden **Elementtyper** och använd knappen  **Egenskaper** för att visa egenskapsdialogrutan.

Det går också att använda skriptgränssnittet för att komma åt registrerade egenskaper och ändra deras värden. I följande exempel antar vi att egenskapen 'weight' är registrerad för den förvalda bågtypen.

```
var nodes = Document.nodes()
for (var i = 0; i < nodes.length; ++i){
    nodes[i].weight = i;
}
for (var i = 0; i < nodes.length; ++i){
    Console.log("weight of node " + i + ": " + nodes[i].weight);
}
```

## 1.3 Handledning

I det här avsnittet ska vi skapa ett exempelprojekt för att utforska några av de viktigaste funktionerna i Rocs. Målet är att skapa en graf och ett skript som åskådliggör en enkel 2-approximativ algoritm för problemet med *minimal hörntäckning*. Minimal hörntäckning är problemet att hitta en delmängd av grafnoder C med minimal storlek, sådan att varje grafbåge är ansluten till minst en nod i C. Problemet är känt som NP-svårt, och vi vill åskådliggöra hur man hittar en approximation med faktorn 2 genom att hitta en motsvarighet i den givna grafen.

Vårt mål är att åskådliggöra förhållandet mellan den matchande och den minimala hörntäckningen. För att göra det behöver vi specificera två bågtyper, en för att visa matchande bågar och en för att visa 'vanliga' bågar, samt två nodtyper som vi använder för att skilja på noder som ingår i C och de som inte ingår i C.

### 1.3.1 Skapa grafen

För att skapa grafen använder vi den förvalda grafgenerering som tillhandahålls av Rocs. Den finns i huvudmenyn under **Grafdokument** → **Verktyg** → **Skapa graf**. Där väljer vi en **Slumpmässig graf** med 30 noder, 90 bågar och fröet 1 (fröet är startfrö för den slumpmässiga grafgenereringen, används samma frö flera gånger erhålles samma och upprepningsbara grafer).

### 1.3.2 Skapa elementtyperna

Vi använder **Elementtyper** och skapar en andra nodtyp samt en andra bågtyp. För båda de nya typerna öppnar vi egenskapsdialogrutan genom att använda de respektive knapparna  **Egenskaper** och ställer in identifierarna till `2`. Dessutom ändrar vi elementens färger för de två nya typerna (för att skilja dem från standardtyperna). Till sist ställer vi in att alla bågar ska vara dubbelriktade, och identifierarna för standardtyperna till `1`.

### 1.3.3 Algoritmen

Till sist måste vi implementera approximationsalgoritmen. Vi använder följande implementering för den:

```
for (var i=0; i < Document.nodes.length; i++) {
    Document.nodes[i].type = 1;
}
for (var i=0; i < Document.edges.length; i++) {
    Document.edges[i].type = 1;
}

var E = Document.edges(); // mängden obehandlade bågar
var C = new Array();      // matchande bågar
while (E.length
> 0) {
    var e = E[0];         // vi väljer första bågen e={u,v}
    var u = e.from();
    var v = e.to();
    e.type = 2;           // ange att bågen är matchande
    E.shift();            // ta bort e (dvs., E[0]) från båglistan
    C.push(u);            // lägg till u i C
    C.push(v);            // lägg till v i C

    // markera u,v som noder i C
    u.type = 2;
    v.type = 2;

    // ta bort alla bågar från E förbundna till u eller v
    var adjacent = u.edges();
    for (var i=0; i < adjacent.length; i++) {
        var index = E.indexOf(adjacent[i]); // hitta index
        if (index != -1) {
            E.splice(index, 1); // ta bort om verkligen hittad
        }
    }
    var adjacent = v.edges();
    for (var i=0; i < adjacent.length; i++) {
        var index = E.indexOf(adjacent[i]); // hitta index
        if (index != -1) {
            E.splice(index, 1); // ta bort om verkligen hittad
        }
    }
}
Console.log("Vertex Cover contains " + C.length + " nodes.");
```

### 1.3.4 Utför algoritmen

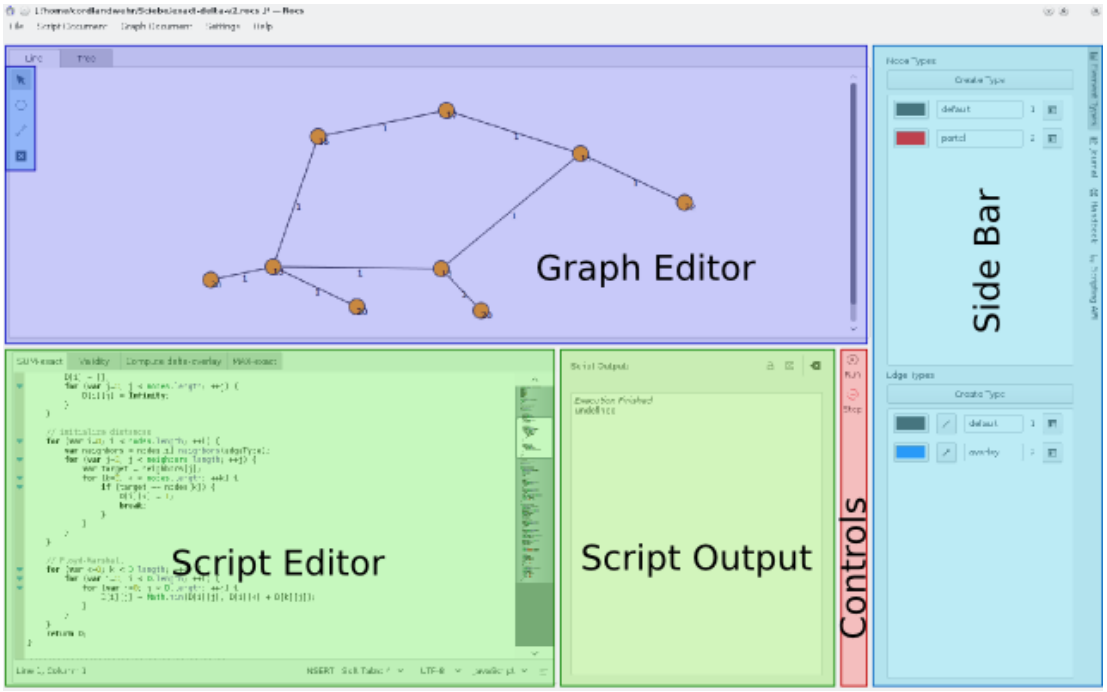
Algoritmen kan beräknas med knappen  **Kör** i skriptkontrollpanelen.

# Kapitel 2

# Rocs användargränssnitt

## 2.1 Användargränssnittets huvuddelar

Användargränssnittet är uppdelat i flera logiska delar som framgår av skärmbilden nedan.




**Grafeditor**

Editorn tillhandahåller en skrivtavla där noder och bågar kan placeras. Ett dubbelklick på något av elementen visar motsvarande egenskapsmeny. Du kan använda verktygen från *sidoradsflikarna* för att skapa och ändra grafer.

Tillgängliga verktyg:

- Längst upp till vänster i rutan finns följande åtgärdsikoner. Att klicka på en åtgärd leder till att muspekaren utför åtgärden på grafeditorns skrivtavla:
  - **Markera och flytta**: För att markera element, klicka antingen på en oanvänd plats på skrivtavlan, håll musknappen nedtryckt och rita en rektangel som innehåller

några noder och/eller bågar för att markera dessa element, eller klicka annars direkt på ett omarkerat element för att markera det. Om du klickar på ett markerat element, eller en uppsättning markerade element, kan de flyttas omkring genom att hålla musknappen nedtryckt och flytta omkring musen. Det är också möjligt att flytta markerade element med piltangenterna.

-  **Lägg till en nod**: Klicka på ett godtyckligt ställe på grafeditorns skrivtavla för att skapa ett nytt dataelement som hör till datastrukturen som för närvarande är markerad. Genom att hålla muspekaren nedtryckt visas en meny där datatypen för det nyskapade dataelementen kan väljas (bara om olika datatyper finns).
-  **Skapa en båge**: Klicka på ett dataelement, håll musknappen nedtryckt och rita en linje till ett annat dataelement, dit bågen ska peka. Åtgärden lyckas bara om den aktuella grafen tillåter att bågen läggs till (i en oriktad graf får t.ex. inte mer än en båge läggas till mellan två dataelement). Genom att hålla muspekaren nedtryckt, dyker en meny upp där bågtypen för de nyskapade bågarna kan väljas (bara om olika bågtyper finns).
-  **Ta bort element**: Klicka på ett element för att ta bort det. Om du tar bort en nod, tas också alla intilliggande bågar bort.

**Sidorad**

Till höger finns sidoraden som tillhandahåller flera verktyg för arbetsflödet:

- **Elementtyper**: Denna grafiska komponent ger direktåtkomst till de tillgängliga båg- och nodtyperna.
- **Journal**: Varje projekt har sin egen journal som t.ex. kan användas för anteckningar om uppgifter, resultat eller observationer.
- **Programmeringsgränssnittet för skript**: Den här grafiska komponenten kan användas för att direkt komma åt handboken och därmed skriptdokumentationen.

**Skripteditor**

I texteditorn kan algoritmer skrivas som förklaras i detalj i kapitel 3. Det går att arbeta med flera skriptdokument samtidigt genom att använda flera flikar.

**Skriptutmatning:**

I det här textområdet visas antingen felsökningsinformation eller skriptutmatning från algoritmer, beroende på val av utmatningsinställning längst upp i den grafiska komponenten. Om ett fel uppstår i skriptet, väljes automatiskt felsökningsutmatning.

**Styrning**

Här hittar man hur körning av algoritmerna styrs. Skriptet som för närvarande är öppet i skripteditorn kan köras med genom att klicka på knappen  **Kör**. Medan skriptet körs, är det möjligt att stoppa det genom att klicka på knappen  **Stopp**.

# Kapitel 3

# Skript

## 3.1 Beräkna algoritmer i Rocs

Rocs använder JavaScript-gränssnittet QtScript internt. Det betyder att alla algoritmer som implementeras måste använda JavaScript. Nedan förklarar vi hur elementen i ett grafdokument kan kommas åt och ändras från skriptgränssnittet. Det är viktigt att observera att ändringar som görs av skriptgränssnittet direkt avspeglas i egenskaperna för grafeditorns element.

### 3.1.1 Styr körning av skript

Det finns olika beräkningssätt för algoritmerna:

-  **Kör**: Kör skriptet till det är klart.
-  **Stoppa**: Stoppa körning av skript (bara tillgängligt medan körning av ett skript pågår).

### 3.1.2 Skriptutmatning

När en algoritm körs visas felsöknings- och programutdata i *felsöknings- och skriptutmatning*. Om skriptgränssnittet upptäcker ett syntaxfel i skriptet visas även det felet som ett felsökningsmeddelande. Observera att alla programmeddelanden också visas i felsökningsutmatningen (med fetstil).

Du kan bestämma vilken text som visas i skriptutmatningen med följande funktioner:

```
Console.log(string message);              // visar meddelandet som ↩
    skriptutdata
Console.debug(string message);            // visar meddelandet som ↩
    felsökningsutdata
Console.error(string message);          // visar meddelandet som ↩
    felutdata
```

### 3.1.3 Programmeringsgränssnittet för skripthantering

De olika delarna av Rocs tillhandahåller var och en ett statiskt element som kan kommas åt av skriptgränssnittet. De är.

- **`Document`** för grafdokumentet
- **`Console`** för loggutmatning på terminalen

För explicit användning av programmeringsgränssnittet och en metodreferens, se direkthjälpen i Rocs sidorad.

# Kapitel 4

# Import och export

## 4.1 Utbyt Rocs-projekt

Rocs-projekt kan importeras och exporteras som arkivfiler `.tar.gz`. Dessa arkiv kan användas för att utbyta projekt. Import och export kan utföras med respektive menyalternativ **Grafdokument** → **Importera graf...** och **Grafdokument** → **Exportera projekt...**.

### 4.1.1 Import och export av grafdokument

Rocs stöder för närvarande import och export av följande filformat:

- DOT-filer, också kända som Graphviz-filer
- GML-filer
- Filer med Trivial Graph-format
- Keyhole Markup Language-format

#### 4.1.1.1 Trivial Graph-filformat

*Trivial Graph-formatet* (TGF) är ett enkelt textbaserat filformat för att beskriva grafer. En TGF-fil består av en lista med noddefinitioner, som avbildar nod-identifierare på etiketter, följt av en lista med bågarna. Det är bara möjligt att ha en etikett per nod och ett värde per båge i detta format. Rocs tolkar importerade grafer som oriktade grafer. Exporterade grafer innehåller två bågar per anslutning om anslutningarna är bidirektionella.

##### 4.1.1.1.1 Formatspecifikation

- Filen börjar med en lista med noder (en nod per rad), följt av en rad med det enda tecknet '#', följt av en lista med bågar (en båge per rad).
- En nod består av ett heltal (identifierare), följt av ett mellanslag, följt av en godtycklig sträng.
- En båge består av två heltal (identifierare) åtskilda med ett mellanslag, följda av ett mellanslag, följt av en godtycklig sträng. Det antas att den riktade bågen pekar från den första identifieraren till den andra.

#### 4.1.1.1.2 Exempel

```
1 startnod
2 sändare
3 mottagare
#
1 2 blå
2 1 röd
2 3 grön
```

### 4.1.1.2 DOT-språket, Graphviz graffilformat

DOT-språket är ett grafbeskrivningsspråk med vanlig text som både tillåter en bra mänskligt läsbar representation av grafer samt en effektiv behandling av graflayoutprogram. DOT är standardfilformat för Graphviz grafvisualiseringssvit, men används också av många andra grafverktyg. De vanliga filändelserna för DOT är `.gv` och `.dot`.

#### 4.1.1.2.1 Funktioner som inte stöds

Rocs kan tolka alla graffiler som innehåller en graf angiven enligt språkdefinitionen för DOT[1]. Stödet för språkets funktioner är fullständigt, trots följande undantag:

- delgraf: På grund av avsaknaden av ett delgrafskoncept i Rocs, importeras bara delgrafer som en uppsättning dataelement och anslutningar. I synnerhet importeras inte anslutningar till eller från delgrafer.
- HTML- och XML-attribut: Attribut (som beteckningar) som innehåller HTML- eller XML-syntax läses oförändrade. I synnerhet läses inte justeringar av teckensnitt och stilar från sådana attribut.

#### 4.1.1.2.2 Exempel

```
digraph min_graf {
    a -> b -> c;
    b -> d;
}
```

[1] https://graphviz.org/doc/info/lang.html

# Kapitel 5

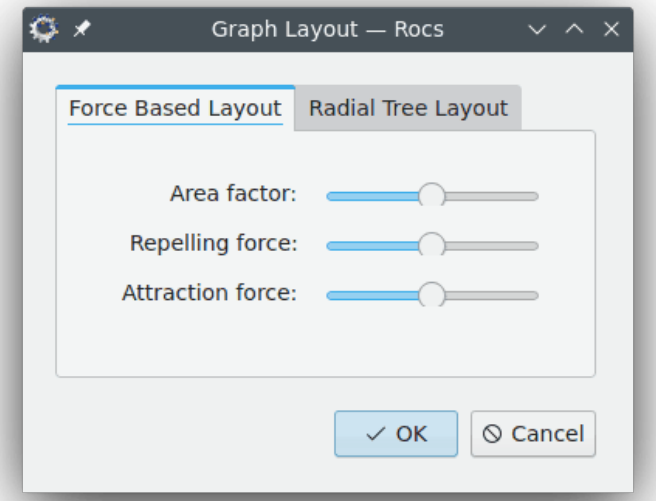
# Graflayout

## 5.1 Automatisk layout av grafer i Rocs

Rocs kan skapa layouter för grafer automatiskt. Rocs graflayoutverktyg finns i huvudmenyn under **Grafdokument** → **Verktyg** → **Graflayout**. Det finns två olika layoutalgoritmer som kan användas: Kraftbaserad layout och Radiell trädlayout. För att använda en av dem, välj motsvarande flik i graflayoutverktyget, välj önskade parametrar och utför algoritmen genom att trycka på knappen **Ok**. Detaljerad information specifik för var och en av layoutalgoritmerna tillhandahålls i följande avsnitt.

### 5.1.1 Kraftbaserad layout

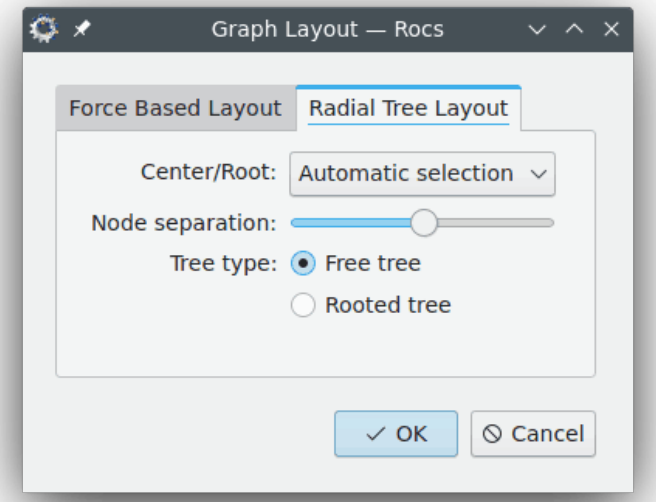
Den kraftbaserade layouten kan användas för vilken graf som helst. Intuitivt simulerar algoritmen krafter som verkar på varje nod. Det finns frånstötande krafter mellan nodpar och attraktiva krafter mellan nodpar som är grannar. Krafternas storlek kan anges genom att flytta motsvarande skjutreglage i användargränssnittet.



En annan parameter som kan styras är Areafaktor. Parametern styr hur noderna sprids ut. Layouter genererade med stora värden på areafaktorn har en tendens att ha större avstånd mellan noder.

#### 5.1.1.1 Radiell trädlayout

Den radiella trädlayouten kan bara användas för träd. Alla försök att använda layoutalgoritmen för andra graftyper producerar ett felmeddelande. Parametrar för den radiella trädlayouten kan väljas genom att använda det tillhandahållna användargränssnittet.




Parametern för trädtyp väljer mellan en fri trädlayout och en rotad trädlayout. I en fri trädlayout placeras noder fritt utan någon uppenbar inbördes hierarki. I en rotad trädlayout placeras rotnoden längst upp och delträd placeras ut under den, vilket ger en idé om hierarkin mellan noder.

Parametern Centrum eller rot definierar vilken nod som kommer att användas som rot för den rotade trädlayouten eller som centrum för den fria trädlayouten. Centrum för den fria trädlayouten är den första noden som placeras ut av algoritmen. Alla andra noder placeras i cirklar omkring centrumnoden. Centrum eller rot kan väljas automatiskt av layoutalgoritmen.

Parametern Nodseparation styr avståndet mellan noderna. Ökas parameterns värde ökar avståndet mellan noder. På liknande sätt, minskas parameterns värde minskas avståndet mellan noder.

# Kapitel 6

# Tack till och licens

Rocs
Program Copyright:

- Copyright 2008 Ugo Sangiori (ugorox SNABELA gmail.com)
- Copyright 2008-2012 Tomaz Canabrava (tcanabrava SNABELA kde.org)
- Copyright 2008-2012 Wagner Reck (wagner.reck SNABELA gmail.com)
- Copyright 2011-2015 Andreas Cord-Landwehr (cordlandwehr SNABELA kde.org)

Dokumentation Copyright:

- Dokumentation copyright 2009 Anne-Marie Mahfouf annma@kde.org
- Dokumentation copyright 2009 Tomaz Canabrava (tcanabrava SNABELA kde.org)
- Dokumentation copyright 2011-2015 Andreas Cord-Landwehr (cordlandwehr SNABELA kde.org)

Översättning Stefan Asserhäll stefan.asserhall@bredband.net