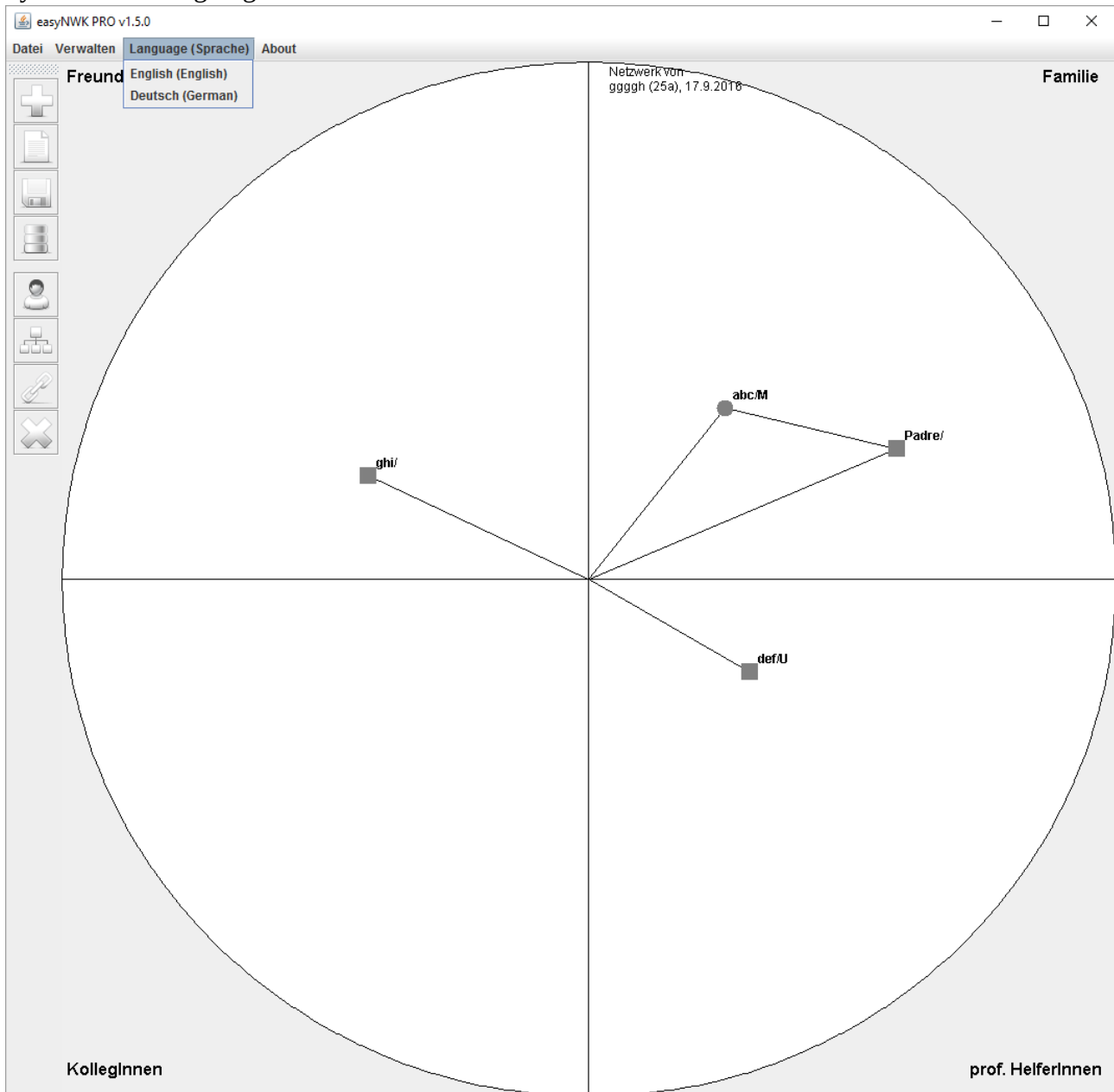


# EasyNWK 1.5.0

## Erklärung des Sprachfeatures

### Die Sprachenauswahl

Mit der Version 1.5 unterstützt die easyNWK jetzt auch Sprachen – dabei sind Deutsch und Englisch statisch (d.h. fix und nur mit Versionsupdates änderbar), während weitere Sprachen dynamisch hinzugefügt werden können.

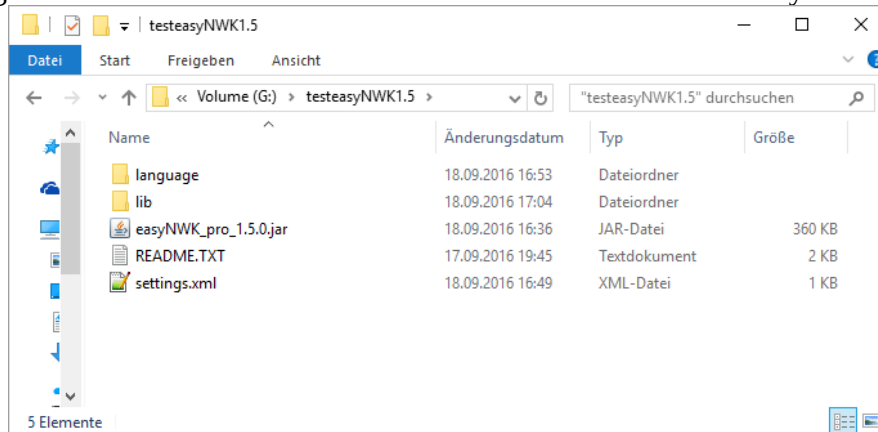


Wie im Screenshot oben ersichtlich, hat sich das Menüpunkt um einen weiteren Punkt erweitert: Language (Sprachen). Unter diesem Menüpunkt sind alle verfügbaren Sprachen aufgelistet. Ein Klick auf die gewünschte Sprache und das User Interface aktualisiert sich von allein mit der neuen Sprache.

## Was hat sich geändert mit dem Versionsupdate

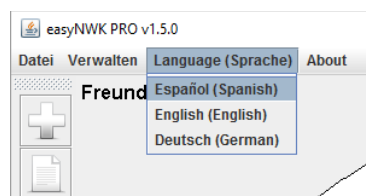
Mit der neuen Version sind teile vom Code stark verändert – damit ist es nicht mehr möglich, nur mehr einfach die easyNWK.jar Datei auszuteilen, da sie jetzt eine Abhängigkeit mit der jsoup Bibliothek hat.

Jsoup ist eine Bibliothek die es erlaubt, einfach auf xml (extended markup language – dazu später mehr) zuzugreifen. Ohne diese Datei startet die Software nicht und sollte daher mit jedem Download mitgeliefert werden! Zu finden ist diese im lib Verzeichnis im easyNWK Ordner.



Die Datei settings.xml und der Ordner language werden von der Software selbst erstellt - falls nicht vorhanden.

Im Ordner language können jene Sprachdateien abgelegt werden, die für die Software erstellt worden sind. Wie diese Dateien ausschauen erkläre ich später. Wird mit der mechanik zum Beispiel die spanische Sprache unterstützt, so wird diese Datei geladen und im Menü entsprechend angezeigt.



Die Datei settings.xml verwaltet die Einstellung mit welcher Sprache gestartet werden soll. Nach dem erstmaligen einstellen kann der Benutzer bei einem erneuten Start in seiner Sprache weiter arbeiten. Falls noch keine Sprache ausgewählt wurde, wird die deutsche Sprache verwendet (unter der annahme, dass die Hauptanwender trotzdem aus dem deutschsprachigen Raum sind).

## Eine neue Übersetzung einbinden

Die Vorlage für die Übersetzung sollte mitgeliefert sein.

XML-Dateien haben den Vorteil, dass man mit ihnen auf Element- und Attributebene arbeiten kann. Ein 'Element' ist dabei immer von einem öffnenden tag (z.B. <translationselement>) und einem schließenden tag (z.B. </translationselement>) eingerahmt. Bei überdachter Struktur lassen sich so einfacher Informationen auslesen und verwenden als bei anderen Formaten. Attribute beschreiben die Elemente, in meinem Fall machen die id's (sprich die Identifikationen) die Unterschiede zwischen den translationselementen aus. Mit id="menuItemData" weiß ich, dass innerhalb dieses Elements die Beschreibung für den ersten Menüpunkt aufgelistet werden.

Lange Erklärung, kurzer Sinn: Für meinem Programm ist der Inhalt der Datei die ideale Struktur zum Arbeiten, für den Übersetzer bietet das Format auch eine Erleichterung da damit auch mehr Informationen dargestellt werden können. In diesem Fall die englische textversion von der Übersetzung.

Die Sprachinformationen haben folgenden Inhalt:

```
<languagemeta>
  <languageshort id="short">de</languageshort>
  <language id="languageName">Deutsch</language>
  <languageinenglish id="languageNameEnglish">
    German</languageinenglish>
</languagemeta>
```

languageshort entspricht der internationalen id. De = Deutsch; En = English; Es = Spanisch; Ru = Russisch; etc.

language ist der Name der Sprache in der jeweiligen Sprache. D.h. Deutsch, English, Español, Français, ...

languageinenglish hilft bei Sprachmangel wenn das Programm in der falschen Sprache eingestellt wurde – zB Program ist auf Spanisch, der Anwender beherrscht die Sprache nicht. Mit der englischen Bezeichnung kann er zumindest erraten welche Sprache verwendet wird. Dieser Punkt ist wichtig wenn arabische Sprachen dazu kommen – oder russisch zum Beispiel.

Im folgenden wird jetzt ein Übersetzungselement erklärt:

```
<translationselement id="menuItemDataNew">
  <english>New</english>
  <translation>Neu</translation>
</translationselement>
```

Die englische Version dient dem Übersetzer als Anhaltspunkt. Dieser Punkt ist nicht zu übersetzen und wird vom Programm auch gar nicht verwendet.

Wichtig ist der 'translation' teil – der Punkt der Übersetzt gehört hab ich grün eingefärbt. Nur hier soll die Übersetzung stehen, der Rest vom Block muss gleich bleiben!

Falls die Struktur beschädigt wird, kann es sein, dass das Programm die Datei nicht lesen kann und deswegen nicht anzeigt. Hier bitte ich also um Vorsicht!

Wird die Datei nun in den Ordner 'language' verschoben, muss nur mehr ein Kritikpunkt erfüllt sein:

Die Datei muss mit der Endung .xml gespeichert sein. Wie sie heißt ist egal, den Namen der Datei verwendet das Program nicht. Wie im Screenshot weiter oben, habe ich also die Datei für die spanische Übersetzung verwendet.

Das Programm übernimmt im laufenden Betrieb nicht automatisch die neue Übersetzung – wieder wenn die Datei aktualisiert worden ist noch wenn die frisch hinein verschoben/kopiert wird. Das heißt: Falls Änderungen hier passieren, bitte das Programm neu starten, dann werden sie auch verwendet.

Es muss nicht gleich alles übersetzt werden: Wenn teile noch in der Ursprungssprache verwendet werden, verwendet das Programm auch diese. D.h. die Übersetzung kann auch nach und nach erfolgen. Wenn translationselement-Teile fehlen, greift das Programm auf die englische Übersetzung zurück.

Fachkundiges Person können also somit die Übersetzungen anbieten, die sie beherrschen. Bei einem guten Plattform (oder Kommunikation), können die Dateien untereinander ausgetauscht und so in größeren Gruppen verwendet werden.