**DIT027  Distributed Fault-tolerant Programming, 3 högskolepoäng**  
Distributed Fault-tolerant Programming, 3 higher education credits  
*Grundnivå / First Cycle*

---

**Fastställande**  

*Utbildningsområde:* Naturvetenskapligt 100 %  
*Ansvarig institution:* Data- och informationsteknik

**Inplacering**  
This course is part of the Software Engineering and Management Bachelor's Programme.

Kursen ingår i följande program: 1) Software Engineering, Bachelor's Programme och 2) Software Engineering and Management, Bachelor's Programme

<table>
<thead>
<tr>
<th>Huvudområde</th>
<th>Fördjupning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Software Engineering</td>
<td>G1F, Grundnivå, har mindre än 60 hp kurs/er på grundnivå som förkunskapskrav</td>
</tr>
<tr>
<td>Tillämpad IT, Software Engineering and Management</td>
<td>G1F, Grundnivå, har mindre än 60 hp kurs/er på grundnivå som förkunskapskrav</td>
</tr>
</tbody>
</table>

**Förkunskapskrav**  
The students must have successfully finished a programming project of 15 higher education credits.

**Mål**  
After completion of the cours the student is expected to be able to.
Kunskap och förståelse
- Explain the process model of Erlang.
- Explain the difference between defensive programming and the let-it-fail approach to handling errors.
- Explain the typical approach to fault tolerance in Erlang.

Färdigheter och förmåga
- Explain the execution and predict the result of running an Erlang program involving pattern matching, (mutually-)recursive functions, concurrent processes and message passing.
- Create skeleton code that can trap errors in other processes using the built-in Erlang functionality as well as the supervisor behavior from Erlang OTP.
- Map a problem domain to an implementation that relies on processes and asynchronous message passing and implement it in Erlang.

Värderingsförmåga och förhållningssätt
- Provide examples of situations where defensive programming is more suited than the let-it-fail approach, and when the opposite is true.

Innehåll
The course provides a solid foundation for programming Erlang, which has all the concepts of distributed fault-tolerant programming. Sequential functional programming, covering items as recursion, are introduced in depth. The students are learning how to write distributed applications, followed by principles of fault-tolerance and design patterns. Programming in Erlang, the students learn about processes, how they communicate and interact. The students also learn about distributed software, and the challenging task of writing fault-tolerant programs. By using Erlang, and OTP, the students are learning how to write applications according to industrial standards, using given design patterns.

Former för undervisning
Lectures and exercises in groups.

Undervisningsspråk: engelska

Former för bedömning
The course is examined by an individual written exam done in an examination hall and programming assignments carried out in groups.
A student who has failed two examinations on the same material has the right to request a change of examiner. Such a request must be submitted to the Department in writing and shall be granted unless there are particular reasons not to do so.

In cases where a course has been discontinued or has undergone major changes, students must be guaranteed at least three examination opportunities (including the regular opportunity) based on the previous content of the course for a period of at least one year.

**Betyg**

På kursen ges något av betygen Väl godkänd (VG), Godkänd (G) och Underkänd (U). In order to be awarded the grade G the student needs to pass both the written exam and the programming assignments. In addition to this, to be awarded the grade VG the student needs to obtain the grade VG on the written exam.

**Kursvärdering**

After completion of the course the students are to be given the possibility of participating in course evaluation anonymously. The processed results of the course evaluations are to be made accessible to students and also made available to new students at the beginning of the next course. Possible changes brought about as a result of course evaluations to be described.

**Övrigt**

Students must arrange with a computer capable of running the software tools provided by the department and are responsible for software installation.

Plagiarism or free-riding is not allowed; cases of such will lead to disciplinary action.