# Detecting deadlocks using static analysis in .NET

Filip Navara

filip.navara@gmail.com

# What did I do last week?

- Implement better resolution of virtual methods using Class Hiearchy Graph
  - Significantly reduced the size of call graph (2,8Mb vs. 4Mb for the LibSEA export)
  - Possible room for more improvement, the fallback code is hit way to often (possibly due to mishandling of generic classes)

# What do I plan to do next week (s)?

- Start adapting the „L.O.V.E.“ prototype
  - Replace the ad-hoc state structures with the HeapObjects and state structure resambling the ones defined by Amy Williams
  - Use data-flow work list algorithm to compute the per-method lock graphs
  - Implement better resolution of virtual methods using Class Hiearchy Graph (currently the type of the called object is not considered in the implementation, but it can significantly reduce the number of possible called methods for eg. System.Object.ToString)
  - Long term: Use the above building block to compute interprocedural lock order graph that takes reentrancy into account